Ahmed Y. Tawfik
Scott D. Goodwin (Eds.)

LNAI 3060

Advances in Artificial Intelligence

**17th Conference of the Canadian Society for Computational Studies of Intelligence, Canadian AI 2004
London, Ontario, Canada, May 2004, Proceedings**

Springer

Lecture Notes in Artificial Intelligence 3060

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

Ahmed Y. Tawfik   Scott D. Goodwin (Eds.)

Advances in Artificial Intelligence

17th Conference of the Canadian Society
for Computational Studies of Intelligence, Canadian AI 2004
London, Ontario, Canada, May 17-19, 2004
Proceedings

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Ahmed Y. Tawfik
Scott D. Goodwin
University of Windsor
School of Computer Science
Windsor, Ontario, N9B 3P4, Canada
E-mail: atawfik@cs.uwindsor.ca;sgoodwin@uwindsor.ca

Library of Congress Control Number: 2004104868

CR Subject Classification (1998): I.2

ISSN 0302-9743
ISBN 3-540-22004-6 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Protago-TeX-Production GmbH
Printed on acid-free paper SPIN: 11007128 06/3142 5 4 3 2 1 0

Preface

Following a long tradition of excellence, the seventeenth edition of the conference of the Canadian Society for the Computational Studies of Intelligence continued the success of its predecessors. This edition reflected the energy and diversity of the Canadian AI community and the many international partnerships that this community has successfully established.

AI 2004 attracted high-quality submissions from Canada and around the world. All papers submitted were thoroughly reviewed by the program committee. Each paper was assigned to at least three program committee members. Out of 105 submissions to the main conference, 29 papers were included as full papers in this volume, and 22 as short/position papers. Three workshops and a graduate symposium were also associated with AI 2004. In this volume, 14 papers selected from 21 submissions to the graduate symposium have been included. We invited three distinguished researchers to give talks representing their active research in AI: Fahiem Bacchus, Michael Littman, and Manuela Veloso.

It would have been impossible to organize such a successful conference without the help of many individuals. We would like to express our appreciation to the authors of the submitted papers, and to the program committee members and external referees who provided timely and significant reviews. In particular, we would like to thank Luis Rueda for organizing the reviewing of the graduate symposium submissions, and Eric Mulvaney for providing valuable assistance in the preparation of the proceedings. To manage the submission and reviewing process we used CyberChair developed by Richard van de Stadt. Christine Günther from Springer has patiently attended to many editorial details. We owe special thanks to Bob Mercer for handling the local arrangements. Last, but not least, we would like to thank the General Chair, Kay Wiese and all the steering committee members for all their tremendous efforts in making AI 2004 a successful conference.

May 2004 Ahmed Y. Tawfik and Scott D. Goodwin

Organization

AI 2004 was organized by the Canadian Society for the Computational Studies of Intelligence (Société Canadienne pour l'Étude de l'Intelligence par Ordinateur).

Executive Committee

Conference Chair	Kay Wiese (Simon Fraser University)
Local Organizer	Bob Mercer (University of Western Ontario)
Program Co-chairs	Ahmed Y. Tawfik (University of Windsor)
	Scott D. Goodwin (University of Windsor)

Program Committee

Aijun An (York U.)
Peter van Beek (U. of Waterloo)
Michael Bowling (U. of Alberta)
Cory Butz (U. of Regina)
Brahim Chaib-draa (U. Laval)
Nick Cercone (Dalhousie U.)
David Chiu (U. of Guelph)
Diane Cook (U. of Texas at Arlington)
Douglas D. Dankel (U. of Florida)
Jim Delgrande (Simon Fraser U.)
Joerg Denzinger (U. of Calgary)
Renée Elio (U. of Alberta)
Richard Frost (U. of Windsor)
Ali Ghorbani (U. of New Brunswick)
Gary Grewal (U. of Guelph)
Jim Greer (U. of Saskatchewan)
Howard Hamilton (U. of Regina)
Bill Havens (Simon Fraser U.)
Graeme Hirst (U. of Toronto)
Michael C. Horsch
(U. of Saskatchewan)
Nathalie Japkowicz (U. of Ottawa)
Froduald Kabanza (U. of Sherbrooke)
Stefan C. Kremer (U. of Guelph)
Amruth Kumar (Ramapo College)
Dekang Lin (U. of Alberta)
Charles Ling (U. of Western Ontario)
Jim Little (U. of British Columbia)
Stan Matwin (U. of Ottawa)
Gord McCalla (U. of Saskatchewan)
Omid Madani (U. of Alberta)
Bob Mercer (U. of Western Ontario)
Evangelos Milios (Dalhousie U.)
Guy Mineau (U. Laval)
Shiv Nagarajan (QNX Systems)
Eric Neufeld (U. of Saskatchewan)
Alioune Ngom (U. of Windsor)
Simon Parsons (Brooklyn College)
Jeff Pelletier (U. of Alberta)
Petra Perner (ibai Leipzig)
David Poole (U. of British Columbia)
Fred Popowich (Simon Fraser U.)
Gregory Provan (Rockwell)
Bob Price (U. of Alberta)
Robert Reynolds (Wayne State U.)
Luis Rueda (U. of Windsor)
Abdul Sattar (Griffith U.)
Dale Schuurmans (U. of Alberta)
Weiming Shen (NRC)
Daniel Silver (Acadia U.)
Bruce Spencer (NRC and UNB)
Suzanne Stevenson (U. of Toronto)
Stan Szpakowicz (U. of Ottawa)
Choh Man Teng (U. of West Florida)
André Trudel (Acadia U.)
Julita Vassileva (U. of Saskatchewan)
Shaojun Wang (U. of Alberta)
Michael Wong (U. of Regina)
Dan Wu (U. of Windsor)
Yang Xiang (U. of Guelph)

Yiyu Yao (U. of Regina)	Kaizhong Zhan (U. of Western Ontario)
Jia You (U. of Alberta)	Nur Zincir-Heywood (Dalhousie U.)
Eric Yu (U. of Toronto)	
Hong Zhang (U. of Alberta)	

Additional Reviewers

Xiangdong An	Zhihua Hu	Gerald Penn
Mohamed Aoun-Allah	Jimmy Huang	M. Shafiei
Julia Birke	Kamran Karimi	Baozheng Shan
Scott Buffett	Vlado Keselj	Yidong Shen
Terry Caelli	Daniel Lemire	Pascal Soucy
Shihyen Chen	Jingping Liu	Finnegan Southey
Lei Duan	Wei Liu	Marius Vilcu
Wael Farag	Yang Liu	Kimberly Voll
Alan Fedoruk	Xiaohu Lu	Xiang Wang
Julian Fogel	Xinjun Mao	Xin Wang
Song Gao	Sehl Mellouli	Kun Wu
P. Gburzynski	Milan Mosny	Qiang Yang
Ali Ghodsi	V. Muthukkumarasamy	Manuel Zahariev
Jasmine Hamdan	Lalita Narupiyakul	Hong Zhang
Malcolm Heywood	Chris Parker	Yan Zhao

Sponsors

National Research Council Canada
Conseil National de Recherches Canada

Canadian Society for the Computational Studies of Intelligence
Société Canadienne pour l'Étude de l'Intelligence par Ordinateur

Table of Contents

Graduate Student Symposium

A Principled Modular Approach to Construct Flexible Conversation Protocols

Roberto A. Flores[1] and Robert C. Kremer[2]

[1] Laval University, Department of Informatics, Sainte-Foy, QC, G1K 7P4, Canada, flores@damas.ift.ulaval.ca
[2] University of Calgary, Department of Computer Science, Calgary, AB, T2N 1N4, Canada
kremer@cpsc.ucalgary.ca

Abstract. Building conversation protocols has traditionally been an art more than a science, as their construction is often guided by designers' intuition rather than by a principled approach. In this paper we present a model for building conversation protocols using inference principles that allow the computational specification and verification of message sequencing and turn-taking. This model, which is based on the negotiation of social commitments, results in highly flexible protocols that support agent heterogeneity while abiding by software engineering practices. We exemplify the specification of protocols using the contract net protocol, a common interaction protocol from the multiagent literature.

1 Introduction

Traditionally, conversations in multiagent systems have been regulated through the use of conversation protocols. More often than not, system designers define these protocols according to the sequences of messages they intuitively believe are best to bring about the actions achieving the goals of their systems. Although such informal approaches free designers of methodological constraints, they reduce protocols to monolithic conversational units with no explicit state properties, a characteristic that limits their implementation in open environments [9][15] and their reuse throughout application domains [1][14]. At the heart of these concerns is the absence of formal principles to build protocols supporting sound software engineering practices (e.g., modularity) as well as designers' autonomy to build heterogeneous agents. Flexible protocols – defined in implementation-independent terms – are needed to achieve seamless interactions between agents programmed using dissimilar techniques and of various levels of sophistication and contextual responsiveness [9]. This versatility requires principles that could be programmed in offline analysis tools (to verify the correctness of protocols at design time) and could also be encoded in deliberative agents as rules (upon which they could infer their most appropriate conversational participation at runtime) [15].

We propose a model to build conversation protocols that fulfill these requirements. This model is based on the notion that conversation protocols aim at the orderly execution of actions, and that responsibilities to perform these actions are established through a series of negotiations to adopt social commitments. In particular, our proposal explicitly indicates the messages allowed (i.e., sequencing) and the agent expected to issue the next message (i.e., turn-taking) in all conversational states.

There have been several recent efforts to define conversation protocols using social commitments, particularly the approaches furthered in [8] and [15]. We share with these approaches the view that message sequencing is reflected through the properties afforded by the progression in the states of communicated commitments. However, these models differ from our proposal in that they fail to formally specify turn-taking as an emergent property of conversational states, and still rely on *ad-hoc* links to indicate the participant advancing the state of commitments at any point in a conversation. Instead, turn-taking in our model flows logically as agents dispose of their obligations (derived from the negotiation of social commitments) by performing both communicative and non-communicative actions. As we detail in this paper, these obligations indicate the types of messages that could be uttered (sequencing) as well as agents expected to issue the next message advancing a conversation (turn-taking), thus defining state properties on which the verification, compilation and execution of protocols can be based. Lastly, our model achieves this functionality while supporting software engineering principles through the modular composition of protocols from reusable components.

The structure of this paper is as follows: the next section is devoted to describing the elements in our model and their underlying principles. This section includes a detailed example of how conversation protocols are defined for simple one-action activities, and a brief explanation on how larger protocols involving several actions can be composed using simpler ones. A subsequent section reviews the characteristics afforded by our model, and discusses how these characteristics allow the modular and reusable composition of flexible protocols and their support for autonomy in open multiagent systems.

2 Modelling Conversations for Action

The notion of *social commitments* [2][11] has been advanced as a way to raise expectations about other agents' performances. Specifically, a social commitment can be defined as an engagement in which an agent is responsible relative to another agent for the performance of an action (independent of whether the agent responsible is also the performer of the action). In our model, social commitments are represented as a three-term predicate of the form

$$\forall\, d, c : \downarrow\!Agent;\ a : \downarrow\!Action \bullet SC(d, c, a)$$

where d and c are agent instances representing the debtor (the agent responsible for the satisfaction of the commitment) and the creditor (the agent on whose behalf the commitment is to be satisfied) of the commitment, and where a is the action that satisfies the commitment. Due to their extent, the description of actions requires a subsequent section of its own.

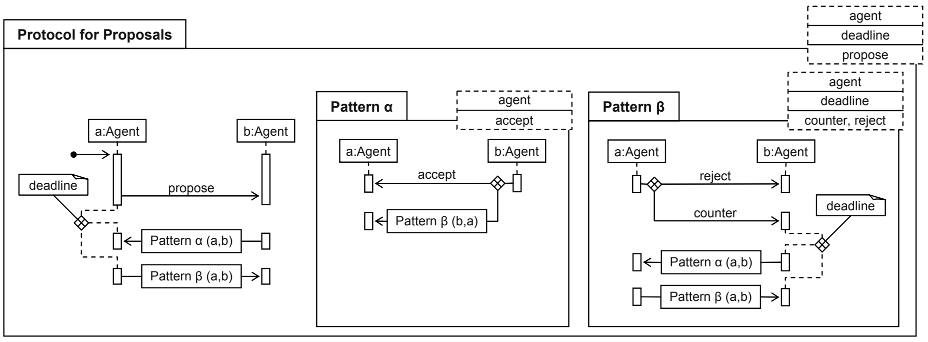


Fig. 1. Interaction diagram of the *protocol for proposals.*

2.1 The Negotiation of Social Commitments

Communicative acts. Inspired by notions from the study of language use [3], we define message interactions as communicative acts from a speaker to an addressee conveying a collection of conversational tokens; and specify the following four tokens to support the negotiation of social commitments:

– *Propose:* to put forth the adoption or discard of a social commitment,
– *Accept:* to accept adopting or discharging a social commitment,
– *Reject:* to reject adopting or discharging a social commitment, and
– *Counter:* to reject a previous proposal while putting forth another proposal to be considered instead.

Lastly, we define a fifth token *Inform* to communicate data.

The protocol for proposals. It is one thing to define communicative acts and quite another to describe how they are used and what they can accomplish in conversations. To that end, we define a negotiations protocol that we call the *protocol for proposals* (*pfp*), which provides a flexible and unambiguous pattern of conversational turn-taking supporting the mutual adoption and discharge of social commitments. As shown in Figure 1, the protocol starts with a proposal from agent a to agent b. This message can be followed (before the expiration of a reply deadline) by the interaction patterns α or β. The interaction pattern α indicates that either agent b sends an accepting message to agent a, or that the interaction continues with pattern β (but with agents a and b's participatory roles inverted, that is, the role of the agent that in pattern α was agent a in pattern β will be agent b, and likewise for agent b). Interaction pattern β indicates that agent a sends a rejection or counterproposal message to agent b, in which case the interaction follows (before the expiration of a reply deadline) by either pattern α or pattern β. In brief, given a proposal from a, b could reply with an acceptance, rejection or counterproposal, or a could issue a rejection or counterproposal to its own proposal. All replies except a counterproposal terminate the instance of the protocol. In theory, a counterproposal can follow another counterproposal *ad infinitum*; in practice, however, successive counterproposals are limited by the reasoning, competence or endurance of interacting agents. Lastly, when an acceptance is issued, both a and b simultaneously apply the proposed (and now accepted) social commitment operation to their record of social commitments, which then affects their record of obligations according to whether they participate as performers in the committed actions.

We specify the *pfp* using three conversation policies:

- *Policy 1:* issuing a proposal (i.e., a message containing a *propose* token) creates an obligation in which the hearer replies with an acceptance, rejection or counterproposal (i.e., a message containing an *accept*, *reject* or *counter* token, respectively) with identical characteristics as those of the proposal.
- *Policy 2:* issuing a reply (i.e., an acceptance, rejection or counterproposal) discards an obligation where the speaker issues a reply.
- *Policy 3:* issuing a proposal followed by a matching acceptance (within proper reply time limits) results in the adoption or discard (whatever operation is involved) of a shared social commitment and the corresponding obligations.

In addition to these policies, we define another policy to raise the expectation that certain social commitments will not remain indefinitely once that they are adopted (thus indicating to the interacting agents that any obligations acquired by adopting the commitment will also be discharged). As such, we define the following additional policy

- *Policy 4:* once a dischargeable social commitment (a subtype of social commitment) has been adopted, there is an agent (usually the one committed to the action) that is responsible for proposing its discharge.

Although the negotiation of social commitments is the central piece upon which the model for conversations is founded, it is nevertheless a vehicle to an end: that of giving rise to social obligations to action. That is, by mutually agreeing to uptake social commitments, agents not only adopt the shared state of these commitments, but also uptake obligations to perform the involved actions (if and only if these agents are specified as the performers of the actions).[1]

2.2 Actions

Figure 2 shows the basic set of action definitions in our model. As indicated in this figure, actions are defined as either individual or composite, where an individual action is an atomic action performed by one agent, and a composite action is a collection of actions performed by one or more agents. Based on the latter, we (simplistically) define joint actions as actions in which there is more

[1] Whether agents abide to their obligations is a matter of study in rational social action (e.g.,[4]), where issues such as norms, trust, reliance, sanctions and reputation influence agents' willingness to comply.

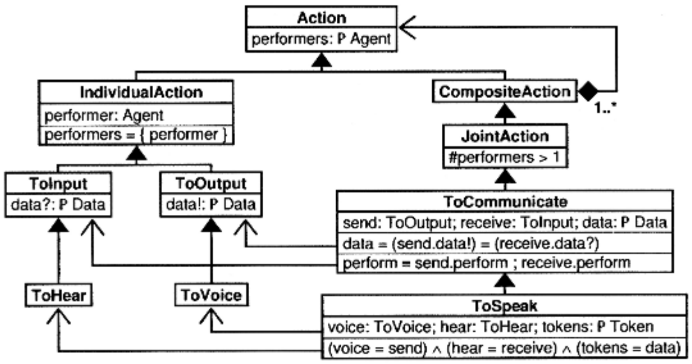

Action
performers: P Agent
IndividualAction
performer: Agent
performers = { performer }
CompositeAction
1..*
JointAction
#performers > 1
ToInput
data?: P Data
ToOutput
data!: P Data
ToCommunicate
send: ToOutput; receive: ToInput; data: P Data
data = (send.data!) = (receive.data?)
perform = send.perform ; receive.perform
ToHear
ToVoice
ToSpeak
voice: ToVoice; hear: ToHear; tokens: P Token
(voice = send) ∧ (hear = receive) ∧ (tokens = data)

Fig. 2. Class diagram of basic actions.

than one performer. The individual actions *ToOutput* and *ToInput* are defined as actions that generate an output and receive an input, respectively. These actions are used in the joint action *ToCommunicate*, which specifies that the output of the *send* action equals the input of the *receive* action, and that the performance of the *send* action precedes that of the *receive* action. Lastly, *ToSpeak* is defined as an action inheriting from *ToCommunicate* in which the *send* and *receive* actions are specialized to the actions *voice* and *hear*, respectively, and where the communicated data is a set of conversational tokens. It is worth mentioning that these and all concepts in the model were defined using the Object-Z specification language [12], which is based on first-order predicate logic, set theory and object-oriented principles. Although in this paper we show a minimum of the model's formal definitions, a more comprehensive account can be found in [5] and [6].

2.3 Creating Protocols: The Contract Net Protocol

The contract net protocol (*cnp*) [13] is a well-know mutual selection strategy in which a manager agent delegates the performance of actions to agents from a set of contender bidding agents. An instance of the protocol begins when a manager sends a request for proposals to agents that could potentially perform these actions. These agents, if willing, submit a proposal to the manager, who then evaluates submitted proposals to select the most suitable candidates to whom to delegate the actions. The protocol terminates when results are send from the delegated agents to the manager. In this section, we show how the principles of our model can be used to define such a conversation protocol between a manager and a bidder. We begin by defining the various data, actions, and agent roles, as well as the joint activity encompassing actions and participants (i.e., agents in specific roles). There are three actions involved in the *cnp*: offering to perform a set of actions (in the form of a proposal), evaluating a set of actions (in the form of an evaluation), and executing a set of actions (in the form of a contract). In this paper, each of these actions and corresponding agent roles are defined independently and then are assembled in a contract net activity, thus illustrating the modular features afforded by our model.

Defining joint actions. The initial peer interaction in the *cnp* is a request from the manager to a bidder to submit a proposal. Abstractly, this interaction can be seen as a request to offer to perform a set of actions.[2] This action is defined as

$$
\begin{array}{|l}
\textit{ToOffer} \\
\hline
\textit{ToProduce} \\
\begin{array}{|l}
\hline
\textit{input}, \textit{output} : \mathbb{P}\ \textit{ConstrainedAction} \\
\hline
(\textit{input} = \textit{in}) \wedge (\textit{output} = \textit{out}) \wedge \\
(\forall\, i, o : \mathbb{P} \downarrow\!\textit{Action} \mid i = \{a : \downarrow\!\textit{Action} \mid \exists\, i_1 : \textit{input} \bullet a = i_1.\textit{action}\} \wedge \\
\qquad\qquad\qquad\quad o = \{a : \downarrow\!\textit{Action} \mid \exists\, o_1 : \textit{output} \bullet a = o_1.\textit{action}\} \bullet o \subseteq i) \\
\hline
\end{array} \\
\hline
\end{array}
$$

where *ToOffer* is a class inheriting from the joint action *ToProduce* (not shown). *ToProduce* defines an individual action receiving an input and producing an output that is followed by a speaking action in which the producer informs the receiver of this output. *ToOffer* shows *input* and *output* as data sets of *ConstrainedAction* items, which are generically specified as an action schema with constraints (not shown), and that the actions in *output* are a subset of the actions in *input*. The action definitions to evaluate (*ToEvaluate*) and to execute actions (*ToExecute*) are defined along the same lines.

Defining agent roles. There are two agent roles involved in a *ToOffer* action: a producer and a receiver. Figure 3 shows the class *OfferRequester* that defines the receiver agent role. This class specifies *RequestingOffer* and *AcceptingOffer* as the utterances that agents of this type are allowed to issue (for simplicity, we omitted other *pfp*-allowed utterances, such as counterproposals and rejections). *RequestingOffer* is modelled as the uttering of a speaking act whose characteristics comply with the specifications in *ToProposeToAdoptOffering* and *ToInformConstrainedActions*, which indicate a message proposing to adopt a commitment to offer, and informing a set of action constraints, respectively. Similarly, *AcceptingOffer* models the uttering of a speaking act compliant with *ToAcceptToDischargeOffering*, which specifies a message accepting to discharge

[2] We see offering as an action whose results are a set of constrained actions that the agent could commit to perform. Just as with any other action (e.g., buying an appliance), results are expected to satisfy qualifying standards (e.g., the appliance satisfies functional and aesthetics requirements); likewise, the standard for results of offering is that the agent could commit to perform the indicated actions. We do not model this type of analysis, as we have purposefully abstracted from the specification of actions (including any standards of quality for their results) to concentrate only on the observable flow of information between agents.

OfferRequester

$$\upharpoonright(\mathit{RequestingOffer}, \mathit{AcceptingOffer})$$

Agent

ToProposeToAdoptOffering

$$a? : \mathit{ToOffer};\ s! : \mathit{ToSpeak}$$

$$\begin{aligned}
&(s!.\mathit{speaker} = a?.\mathit{receiver} = \mathit{self}) \land (s!.\mathit{addressee} = a?.\mathit{producer}) \land \\
&(\exists\, p : \mathit{Propose} \mid p = \mathit{ProposeToAdoptOffering}(a?) \land \\
&\qquad p.\mathit{reply}.\mathit{until} \geq \mathit{now} \bullet p \in s!.\mathit{tokens})
\end{aligned}$$

ToAcceptToDischargeOffering

$$a? : \mathit{ToOffer};\ s! : \mathit{ToSpeak}$$

$$\begin{aligned}
&(s!.\mathit{speaker} = a?.\mathit{receiver} = \mathit{self}) \land (s!.\mathit{addressee} = a?.\mathit{producer}) \land \\
&(\exists\, a : \mathit{Accept} \mid a = \mathit{AcceptToDischargeOffering}(a?) \bullet a \in s!.\mathit{tokens})
\end{aligned}$$

ToInformConstrainedActions

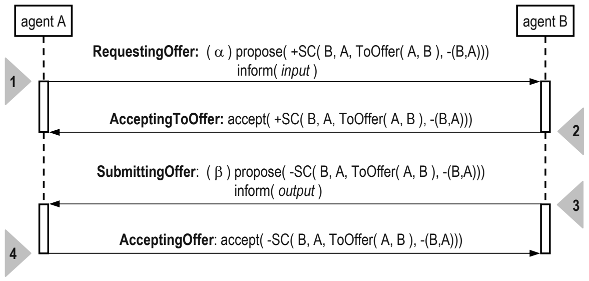
$$\mathit{input}? : \mathbb{P}\,\mathit{ConstrainedAction};\ a? : \mathit{ToOffer};\ s! : \mathit{ToSpeak}$$

$$\begin{aligned}
&(a?.\mathit{input} = \mathit{input}?) \land \\
&(\exists\, i : \mathit{Inform} \mid i = \mathit{InformConstrainedActions}(\mathit{input}?) \bullet i \in s!.\mathit{tokens})
\end{aligned}$$

$$\begin{aligned}
&\mathit{RequestingOffer} \mathrel{\widehat=} \mathit{ToProposeToAdoptOffering} \bigwedge \\
&\qquad \mathit{ToInformConstrainedActions} \mathbin{⨾} \mathit{SendUtterance} \\
&\mathit{AcceptingOffer} \mathrel{\widehat=} \big[\, a? : \mathit{ToOffer} \mid (a?.\mathit{receiver} = \mathit{self}) \land \\
&\qquad \mathit{ReplyToProposeToDischargeOffering}(a?.\mathit{receiver}, \\
&\qquad\qquad a?.\mathit{producer}, a?) \in \operatorname{dom} \mathit{obligations} \,\big] \bullet \\
&\qquad \mathit{ToAcceptToDischargeOffering} \mathbin{⨾} \mathit{SendUtterance}
\end{aligned}$$

Fig. 3. Definition of the *OfferRequester* agent role.

the commitment to offer, and defines as a precondition that the speaker has an obligation to reply to a proposal to discharge the commitment to offer (as specified in *ReplyToProposeToDischargeOffering*, which is not shown). The counterpart agent role *OfferProvider* (not shown) is defined similarly. It specifies the utterances *AcceptingToOffer*, *RejectingToOffer* and *SubmittingOffer*, where the first accepts and the second rejects to adopt a commitment to offer (if and only if there is the obligation to reply to a proposal to add a commitment to offer), and where the third informs a set of actions and proposes to discharge a commitment to offer (if and only if there is the obligation to propose to discharge this commitment). A fourth utterance *AcceptingandSubmittingOffer* is defined as a shortcut for cases in which an acceptance to adopt and a proposal to discharge committing to offer (along with the resulting set of actions) are


agent A
agent B
RequestingOffer: (α) propose(+SC(B, A, ToOffer(A, B), -(B,A)))
inform(input)
AcceptingToOffer: accept(+SC(B, A, ToOffer(A, B), -(B,A)))
SubmittingOffer: (β) propose(-SC(B, A, ToOffer(A, B), -(B,A)))
inform(output)
AcceptingOffer: accept(-SC(B, A, ToOffer(A, B), -(B,A)))
1
2
3
4

	A & B's Shared Commitments		A's Obligations		B's Obligations
State 1		policy 1: add 1	1. ToSpeak(B→A,{ReplyTo(α)})	policy 1: add 1	1. ToSpeak(B→A,{ReplyTo(α)})
		policy 1: add 2	2. ToHear(→A,{ReplyTo(α)})	policy 1: add 2	2. ToVoice(B→,{ReplyTo(α)})
State 2	policy 3: add Z	policy 2: delete 1	~~1. ToSpeak(B→A,{ReplyTo(α)})~~	policy 2: delete 1	~~1. ToSpeak(B→A,{ReplyTo(α)})~~
	Z. (B,A,ToOffer(B→A))	policy 2: delete 2	~~2. ToHear(→A,{ReplyTo(α)})~~	policy 2: delete 2	~~2. ToVoice(B→,{ReplyTo(α)})~~
		policy 3: add 3	3. ToOffer(B→A)	policy 3: add 3	3. ToOffer(B→A)
		policy 3: add 4	4. ToSpeak(B→A,{Inform(OUTPUT)})	policy 3: add 4	4. ToProduce(B,INPUT,OUTPUT)
		policy 3: add 5	5. ToHear(→A,{Inform(OUTPUT)})	policy 3: add 5	5. ToSpeak(B→A,{Inform(OUTPUT)})
		policy 4: add 6	6. ToSpeak(B→A,{Propose(-Z)})	policy 3: add 6	6. ToVoice(B→,{Inform(OUTPUT)})
		policy 4: add 7	7. ToHear(→A,{Propose(-Z)})	policy 4: add 7	7. ToSpeak(B→A,{Propose(-Z)})
				policy 4: add 8	8. ToVoice(B→,{Propose(-Z)})
State 3	Z. (B,A,ToOffer(B→A))	policy 1: add 8	3. ToOffer(B→A)	policy 1: add 9	3. ToOffer(B→A)
		policy 1: add 9	4. ToSpeak(B→A,{Inform(OUTPUT)})	policy 1: add 10	4. ToProcess(B,INPUT,OUTPUT)
			5. ToHear(→A,{Inform(OUTPUT)})		5. ToSpeak(B→A,{Inform(OUTPUT)})
			6. ToSpeak(B→A,{Propose(-Z)})		6. ToVoice(B→,{Inform(OUTPUT)})
			7. ToHear(→A,{Propose(-Z)})		7. ToSpeak(B→A,{Propose(-Z)})
			8. ToSpeak(A→B,{ReplyTo(β)})		8. ToVoice(B→,{Propose(-Z)})
			9. ToVoice(A→,{ReplyTo(β)})		9. ToSpeak(A→B,{ReplyTo(β)})
					10. ToHear(→B,{ReplyTo(β)})
State 4	policy 3: delete Z	policy 2: delete 8	~~3. ToOffer(B→A)~~	policy 2: delete 9	~~3. ToOffer(B→A)~~
	~~Z. (B,A,ToOffer(B→A))~~	policy 2: delete 9	~~4. ToSpeak(B→A,{Inform(OUTPUT)})~~	policy 2: delete 10	~~4. ToProcess(B,INPUT,OUTPUT)~~
		policy 3: delete 3	~~5. ToHear(→A,{Inform(OUTPUT)})~~	policy 3: delete 3	~~5. ToSpeak(B→A,{Inform(OUTPUT)})~~
		policy 3: delete 4	~~6. ToSpeak(B→A,{Propose(-Z)})~~	policy 3: delete 4	~~6. ToVoice(B→,{Inform(OUTPUT)})~~
		policy 3: delete 5	~~7. ToHear(→A,{Propose(-Z)})~~	policy 3: delete 5	~~7. ToSpeak(B→A,{Propose(-Z)})~~
		policy 4: delete 6	~~8. ToSpeak(A→B,{ReplyTo(β)})~~	policy 3: delete 6	~~8. ToVoice(B→,{Propose(-Z)})~~
		policy 4: delete 7	~~9. ToVoice(A→,{ReplyTo(β)})~~	policy 4: delete 7	~~9. ToSpeak(A→B,{ReplyTo(β)})~~
				policy 4: delete 8	~~10. ToHear(→B,{ReplyTo(β)})~~

Fig. 4. Interaction diagram and states of obligations and commitments.

issued in one message. This utterance is defined with the same preconditions as *AcceptingToOffer* and *RejectingToOffer*. Similarly, the agent roles *EvaluationRequester*, *EvaluationProvider*, *ExecutionRequester* and *ExecutionProvider* define utterances advancing conversations for the actions to evaluate and to execute actions, respectively.

The dynamics of interaction. Preconditions indicate whether an utterance could be issued at any given state in a conversation. In the case of the agent roles in the offering action, all utterances are defined with preconditions except for the requester's *RequestingOffer*. The absence of preconditions is not necessarily an indicator that an utterance could initiate a conversation; a precondition for this utterance could have been, for example, that a commitment with the same characteristics has not already been adopted. State transitions in conversations occur when an utterance makes one or more of the *pfp* conversation policies alter the obligations of agents.

In this section, we illustrate the dynamics of inference in our model, in which utterances cause transitions to new states (indicating obligations to speak) that enable utterances that cause advances to further states and obligations.[3] Figure 4 shows a conversation to offer between agents A and B. The initial assumption in this example is that A (the *OfferRequester*) and B (the *OfferProvider*) do not have any obligations enabling utterances other than A's *RequestingOffer*. Once uttered, this proposal (labelled α in the figure) triggers policy 1 (the uttering of a proposal creates obligations to reply to it), which establishes obligations where B performs the voicing and A performs the hearing in a speaking joint action replying to α (shown in state 1 as A and B's obligations 1 and 2). By modifying A and B's obligations, this policy effectively changed the state of their conversation, enabling some utterances (B's *AcceptingToOffer*, *Rejecting-ToOffer* and *AcceptingandSubmittingOffer*) while disabling others (in this case, none). Since these enabled utterances have the characteristics that make them adequate replies to α, it is expected that B will utter one of them to satisfy his obligations. We show the case in which B utters *AcceptingToOffer*, causing a transition to state 2 through the triggering of the following policies: policy 2 (once a reply is uttered it discharges obligations to reply), which discharges A and B's obligations 1 and 2; policy 3 (given a proposal and a matching acceptance, agents apply the negotiated commitment operation), which results in the adoption of social commitment *Z*, A's obligations 3 to 5 (indicating that she jointly participates with B in a *ToOffer* and a *ToSpeak* joint actions in which she performs a *ToHear* individual action) and B's obligations 3 to 6 (indicating that he participates with A in the same *ToOffer* and *ToSpeak* joint actions although he performs a *ToProcess* and a *ToVoice* individual actions); and policy 4 (adopting a dischargeable commitment creates obligations to propose its discharge), which creates A's obligations 6 and 7, and B's obligations 7 and 8, which indicate that B is ought to propose to A discharging commitment *Z*. Reaching this state enables B's utterance *SendingOffer* while disabling all others (except the initial request, which is always enabled). State 3 shows the resulting set of commitments and obligations after B's issuing of a *SubmittingOffer* message that proposes to discharge commitment *Z* (labelled β) and informs an offer. This utterance triggers policy 1, which results in A's obligations 8 and 9 and B's obligations 9 and 10 indicating that these agents are the voicer and hearer, respectively, in a speaking action replying to proposal β. The only enabled utterance in this state is A's *AcceptingOffer*. As shown in state 4, this utterance triggers policy 2, which discharges obligations to reply to β (A's obligations 8 and 9, B's obligations 9 and 10); policy 3, which discharges commitment *Z* and corresponding obligations (A's obligations 3 to 5; B's obligations 3 to 6); and policy 4, which discharges obligations to propose discharging *Z* (A's obligations 6 and 7; B's obligations 7 and 8). At this point, no further obligations to speak remain thus signaling the end of the conversation.

[3] Readers interested in a formal account of the dynamics of conversations can refer to [5], where a segment of a conversation similar to the one hereby presented is analyzed through strict logical proofs.

Manager

$$\upharpoonright(\mathit{RequestingBid}, \mathit{AcceptingToEvaluate}, \mathit{AwardingContract},\\ \mathit{AcceptingToEvaluateandAwardingContract}, \mathit{RejectingBid},\\ \mathit{AcceptingToEvaluateandRejectingBid}, \mathit{AcceptingResults})$$

$$\mathit{OfferRequester}\ \mathit{EvaluationProvider}\ \mathit{ExecutionRequester}$$

$$\mathit{RequestingBid} \mathrel{\widehat{=}} \mathit{RequestingOffer}$$

$$\begin{aligned}
&\mathit{AcceptingToEvaluate} \mathrel{\widehat{=}} \big[\, a? : \mathit{ToOffer};\ e? : \mathit{ToEvaluate} \,\big|\\
&\quad (a?.\mathit{receiver} = e?.\mathit{producer} = \mathit{self}) \land (a?.\mathit{producer} = e?.\mathit{receiver}) \land\\
&\quad \{\mathit{ReplyToProposeToDischargeOffering}(a?.\mathit{receiver}, a?.\mathit{producer}, a?),\\
&\quad\ \ \mathit{ReplyToProposeToAdoptEvaluating}(e?.\mathit{producer},\\
&\qquad\qquad\qquad\qquad e?.\mathit{receiver}, e?)\} \subseteq \operatorname{dom} \mathit{obligations} \,\big] \bullet\\
&\quad \mathit{ToAcceptToDischargeOffering} \land\\
&\quad \mathit{ToAcceptToAdoptEvaluating} \ {}_{9}^{\circ}\ \mathit{SendUtterance}
\end{aligned}$$

$$\begin{aligned}
&\mathit{AwardingContract} \mathrel{\widehat{=}} \big[\, e? : \mathit{ToEvaluate};\ x? : \mathit{ToExecute} \,\big|\\
&\quad (e?.\mathit{producer} = x?.\mathit{receiver} = \mathit{self}) \land (e?.\mathit{receiver} = x?.\mathit{producer}) \land\\
&\quad \mathit{SpeakToProposeToDischargeEvaluating}(e?.\mathit{producer},\\
&\qquad\qquad\qquad\qquad e?.\mathit{receiver}, e?) \in \operatorname{dom} \mathit{obligations} \,\big] \bullet\\
&\quad \mathit{ToProposeToDischargeEvaluating} \land \mathit{ToInformEvaluation} \land\\
&\quad \mathit{ToProposeToAdoptExecuting} \land \mathit{ToInformConstrainedActions} \ {}_{9}^{\circ}\\
&\quad \mathit{SendUtterance}
\end{aligned}$$

Definition *AcceptingToEvaluateandAwardingContract* omitted.
Definition *AcceptingToEvaluateandRejectingBid* omitted.

$$\mathit{RejectingBid} \mathrel{\widehat{=}} \mathit{SubmittingEvaluation}$$

Fig. 5. Partial definition of the *Manager* agent role.

Defining joint activities. Joint activities are the components where actions and agent roles are integrated. In this example, we define a contract net joint activity in which a manager and a bidder interact. Figure 5 shows the definition of the agent role *Manager*, which inherits from the agent roles *OfferRequester*, *EvaluationProvider* and *ExecutionRequester*. This agent role defines several utterances. *RequestingBid* is defined as *OfferRequester*'s utterance *RequestingOffer*. *AcceptingToEvaluate* is an utterance accepting the discharge of a commitment to offer, and accepting the adoption of a commitment to evaluate; this can be uttered only when there are obligations to reply to a proposal to discharge a commitment to offer, and a proposal to adopt a commitment to evaluate. *AwardingContract* is an utterance proposing the discharge of and the informing of the results of evaluating, and proposing to adopt the execution of informed actions; this can be uttered only when there is an obligation to propose the discharge of a commitment to evaluate. *AcceptingToEvaluateandAwardingContract* (not shown) is an utterance encompassing *AcceptingToEvaluate* and *Awarding-*

Contract. *RejectingBid* is *EvaluationProvider*'s utterance *SubmittingEvaluation*. *AcceptingToEvaluateandRejectingBid* (not shown) is an utterance encompassing *AcceptingToEvaluate* and *RejectingBid*. Finally, *AcceptingResults* is an utterance inherited unchanged from agent role *ExecutionRequester*. The agent role *Bidder* (not shown) is defined in a similar vein as in *Manager*.

The joint activity *ContractNet*, which encompasses the aforementioned definitions, is shown in Figure 6. This class defines *manager* and *bidder* as the participants, *requirements*, *bid*, *evaluation* and *contract* as the data sets, and *bidding*, *evaluating* and *executing* as the actions in the activity. This class also specifies the roles that the *manager* and *bidder* play in each of these actions, the relationship between the data sets and the input and output variables in the actions, and that these actions are the only actions in the activity. Lastly, a protocol (which is also shown in Figure 7) is defined as the sequential arrangement of utterances that participants could issue according to their role types. Although not shown in this paper, these sequences conform to the strict dependency between utterances and obligations that give way to state transitions such as the one exemplified through the interaction in the previous section.

2.4 Flexibility

We study flexibility in terms of the number of unique message sequences that could be supported by a protocol. In the case of our model, flexibility is mainly afforded by the ability of agents to issue counterproposals. As such, to calculate the number of unique message sequences, we define $\kappa : \mathbb{N}_1$ as the maximum number of consecutive counterproposals that could be issued in a *pfp* instance. For example, a κ of 1 indicates that any counterproposal is automatically rejected; a κ of 2 indicates that the second consecutive counterproposal is automatically rejected, and so on. Since it effectively limits the maximum number of messages in a sequence, we call κ the depth of a *pfp* instance. Using κ, we define the number of unique message sequences ending in an acceptance and ending in a rejection (denoted by A and R, respectively, in Formula 1) for any *pfp* instance.

$$A = \sum_{i=1}^{\kappa} 2^{i-1} \qquad R = 2^{\kappa} + \sum_{i=1}^{\kappa} 2^{i} \tag{1}$$

Conversations are composed of more than one consecutive *pfp* sequence. As such, we define $\eta : \mathbb{N}_1$ as the number of consecutive *pfp* instances in a protocol. Using η, we can calculate the number of consecutive *pfp* instances ending in acceptances (denoted by N in Formula 2), and the total number of unique message sequences in a series of η consecutive *pfp* instances (denoted by T in Formula 3).

$$N = \mathbb{A}^{\eta-1} * A \qquad \mathbb{A} = \frac{3A+1}{2} \tag{2}$$

$$T = R_1 + \mathbb{A}_1(R_2 + \mathbb{A}_2(...R_{\eta-1} + \mathbb{A}_{\eta-1}(R_\eta + A_\eta)...)) \tag{3}$$

ContractNet

JointActivity

$$
\begin{array}{l}
manager : \downarrow Manager;\ bidder : \downarrow Bidder;\ requirements, bid : \mathbb{P}_1\ ConstrainedAction \\
evaluation : \mathbb{P}_1\ EvaluationItem;\ contract : \mathbb{P}_1\ ConstrainedAction \\
bidding : ToOffer;\ evaluating : ToEvaluate;\ executing : ToExecute \\
\hline
(manager = bidding.receiver = evaluating.producer = executing.receiver) \land \\
(bidder = bidding.producer = evaluating.receiver = executing.producer) \land \\
(requirements = bidding.input) \land (bid = bidding.output = evaluating.input) \land \\
(evaluation = evaluating.output) \land (contract = executing.input) \land \\
(\forall\, e, c : \mathbb{P} \downarrow Action \mid e = \{a : \downarrow Action \mid \exists\, e_1 : evaluation \bullet a = e_1.action\} \land \\
\qquad c = \{a : \downarrow Action \mid \exists\, c_1 : contract \bullet a = c_1.action\} \bullet c \subseteq e) \land \\
(\forall\, a_1, a_2, a_3 : \downarrow JointAction \mid a_1 = bidding \land a_2 = evaluating \land a_3 = executing \\
\quad \bullet\ actions = \{a_1, a_2, a_3\})
\end{array}
$$

$$
\begin{array}{l}
Protocol \mathrel{\widehat=} [\, m? : \downarrow Manager;\ b? : \downarrow Bidder \mid m? = manager \land b? = bidder \,] \bullet \\
m?.RequestingBid \fatsemi (b?.RejectingToBid\ [] \\
(((b?.AcceptingToBid \fatsemi b?.SubmittingBid)\ [] \\
\quad b?.AcceptingToBidandSubmittingBid) \fatsemi \\
\quad ((((m?.AcceptingToEvaluate \fatsemi m?.AwardingContract)\ [] \\
\qquad m?.AcceptingToEvaluateandAwardingContract) \fatsemi \\
\qquad (((b?.AcceptingAward \fatsemi b?.SubmittingResults)\ [] \\
\qquad\quad b?.AcceptingAwardandSubmittingResults) \fatsemi m?.AcceptingResults))\ [] \\
\quad (((m?.AcceptingToEvaluate \fatsemi m?.RejectingBid)\ [] \\
\qquad m?.AcceptingToEvaluateandRejectingBid) \fatsemi b?.AcceptingBidRejection))))
\end{array}
$$

Fig. 6. Definition of the *Contract Net* joint activity.

We made several assumptions to simplify our evaluation of flexibility[4]: a) a successful protocol is one that follows a sequence of *pfp* instances where each instance terminates with an acceptance; b) interacting agents alternate as initiators of *pfp* instances (e.g., the manager's proposal to adopt committing to submit a bid is followed by the bidder's proposal to discharge the commitment to submit a bid); and c) in cases where the agent issuing an acceptance is the same that is expected to issue the proposal in the subsequent *pfp* instance, this agent has the option of uttering these messages separately or as one message (e.g., the manager's acceptance to commit to evaluate a bid and his proposal to discharge this commitment could be uttered as one message). Given the latter assumption, we redefine the number of possible accepting messages in a non-terminating *pfp* instance (i.e., all *pfp* instances i in a sequence of η *pfp* instances,

[4] Although these assumptions reduce the number of possible sequences supported by the *pfp*, it facilitates calculating sequences in protocols abiding to them.

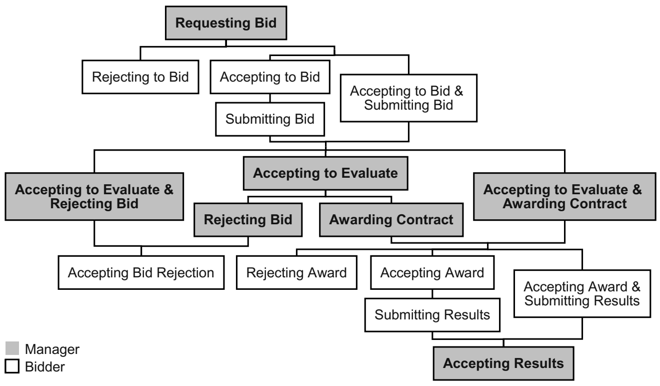


Fig. 7. Protocol in the *Contract Net* joint activity.

where $i : \mathbb{N}_1 \bullet i \leq \eta$) as $\mathbb{A}$ in Formula 2. The rationale for this formula is that there are A possible acceptances in a *pfp* instance of depth κ, where $\frac{A-1}{2}$ of these acceptances could be issued by the agent that uttered the initial proposal, and where the remaining $\frac{A+1}{2}$ are acceptances that could be issued by the agent who was initially proposed. Given our assumption that the agent who was proposed will utter the next *pfp*'s proposal indicates that he could either issue $\frac{A+1}{2}$ acceptances followed by a separate proposal, or that he could issue $\frac{A+1}{2}$ messages encompassing both an acceptance an a proposal. Thus, the possible number of unique message sequences terminating in an acceptance for a non-terminating *pfp* instance is the number of sequences in which the proposing agent utters the acceptance, plus the number of sequences in which the proposed agent utters a message containing only the acceptance, plus the number of sequences in which this same agent utters the acceptance along with the subsequent proposal.

These formulas help us calculate the number of unique message sequences in the *cnp*, and evaluate these results against a comparable *ad-hoc* protocol, e.g., the FIPA contract net protocol [7]. In brief, this latter protocol specifies an initiator (a manager) issuing a call for proposals to participants (bidders), who can either refuse (thus terminating the protocol) or propose to the initiator. This propose can then be followed by a rejection (terminating the protocol) or acceptance from the initiator, and then by a failure or an inform (indicating success), any of which signals the termination of the protocol (in fact, there could be an inform indicating results or an inform indicating successful execution; given the minimal semantic differences between these messages, we hereby consider them as one). In the case of our *cnp* definition, which has $\eta = 4$, and assuming a minimum flexibility κ=1 (i.e., no counterproposals allowed), we derive (from Formula 2) that there exist 8 unique sequences leading to a successful termination, and (from Formula 3) that there are 68 unique sequences in total. If one counterproposal was allowed (κ=2), these numbers would increase to 375 and 1,935 sequences, respectively. These results contrast with the 4 possible unique sequences in the FIPA contract net protocol, of which only one leads to a successful termination. In addition, we achieve this flexibility without significant message overhead: while the number of messages for a successful termination in the FIPA request is 4, in our model is a minimum of 5. From the aforementioned, we could conclude that our model for conversations effectively supports more flexible conversations, without a significant overhead, than those afforded by *ad-hoc* conversations protocols.

3 Conclusions

It has been mentioned in recent years that the key desirable criteria for models for conversation protocols is their support for agent heterogeneity (e.g., [9][15]) and the definition of principles for their modular composition and re-use (e.g., [1][14]). In this paper we presented a model for conversations that supports this criteria by specifying declarative principles that allow building protocols in a modular manner. On the one hand, this model supports heterogeneity by defining protocols through social commitment messages yielding obligations influencing action performances. In particular, by focusing on the properties of messages rather than on the implementation of actions, it allows designers to implement agents using their programming technique of choice; by featuring an inferential definition of sequencing and turn-taking, it allows protocols whose correctness could be verified at design time and then hard-coded in agents using a procedural language or programmed as inference rules in deliberative agents (whom then could verify compliance to the protocol at runtime); by supporting counterproposals it allows flexible protocols that sophisticated agents can use to exploit context-dependent circumstances, while also allowing interactions with less-able agents that only follow message sequences leading directly to successful terminations (and whom may not pursue or immediately reject counterproposals). In addition, the principles in the model are compliant to meaningful, verifiable and declarative criteria allowing for the compilation (at design time) and execution (at runtime) of protocols, as defined in [15]. These principles also support agent autonomy by negotiating rather than imposing the uptake of communicated commitments. On the other hand, our model supports compositional principles by defining protocols through modular components defined in a formal object-oriented specification language. In particular, the model supports the five criteria for modularity defined in [10]: it allows decomposability by structuring protocols in terms of loosely coupled components such as obligations, joint actions, agent roles, and so on; it allows composability by reusing such components to build new protocols through inheritance and aggregation; it allows understandability by abstracting the details of message, action and role definitions in the specification of protocols; it allows continuity by enabling the refinement and extension of protocols with minimal impact on their structural components; and lastly, it allows protection by defining inference principles upon which the verification of and compliance to protocols can be rigorously analyzed both at design and runtime.

Acknowledgements. We are thankful for support from the National Science and Engineering Research Council (NSERC) of Canada, and the Alberta Software Engineering Research Consortium (ASERC). We are also grateful to the anonymous reviewers for their valuable suggestions and comments.

References

1. S. Bussmann, N.R. Jennings, and M.J. Wooldridge. Re-use of interaction protocols for decision-oriented applications. In P. Ciancarini and M. Wooldridge, editors, *Proceedings of the 3rd International Workshop on Agent-Oriented Software Engineering*, pages 51–62, July 2002.
2. C. Castelfranchi. Commitments: From individual intentions to groups and organizations. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 41–48, San Francisco, CA, June 1995.
3. H.H. Clark. *Using language.* Cambridge University Press, 1996.
4. R. Conte and C. Castelfranchi. *Cognitive and Social Action.* University College London Press, 1995.
5. R.A. Flores. *Modelling agent conversations for action.* PhD thesis, Department of Computer Science, University of Calgary, June 2002.
6. R.A. Flores and R.C. Kremer. To commit or not to commit: Modelling agent conversations for action. *Computational Intelligence*, 18(2):120–173, 2003.
7. Foundation for Intelligent Physical Agents (FIPA). FIPA interaction protocol specifications. `http://www.fipa.org/repository/ips.php3`, October 2003.
8. N. Fornara and M. Colombetti. Defining interaction protocols using a commitment-based agent communication language. In J.S. Rosenschein, T. Sandholm, M.J. Wooldridge, and M. Yokoo, editors, *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 520–527, Melbourne, Australia, July 2003.
9. M. Greaves, H. Holmback, and J.M. Bradshaw. What is a conversation policy? In M. Greaves and J.M. Bradshaw, editors, *Proceedings of the Workshop on Specifying and Implementing Conversation Policies*, pages 1–9, Seattle, WA, 1999.
10. B. Meyer. *Object-Oriented Software Construction.* Prentice Hall, second edition, 1997.
11. M.P. Singh. Social and psychological commitments in multiagent systems. In *AAAI Fall Symposium on Knowledge and Action at Social and Organizational Levels*, Monterey, California, November 1991.
12. G. Smith. *The Object-Z Specification Language.* Kluwer Publishers, 2000.
13. R.G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, 29(12):1104–1113, 1980.
14. B. Vitteau and M.-P. Huget. Modularity in interaction protocols. In F. Dignum, editor, *Advances in Agent Communication*, volume 2922 of *Lecture Notes in Artificial Intelligence*, pages 291–309. Springer Verlag, 2004.
15. P. Yolum and M.P. Singh. Flexible protocol specification and execution: Applying event calculus planning using commitments. In C. Castelfranchi and W.L. Johnson, editors, *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 527–534, Bologna, Italy, July 2002.

Balancing Robotic Teleoperation and Autonomy for Urban Search and Rescue Environments

Ryan Wegner and John Anderson

Autonomous Agents Laboratory
Department of Computer Science
University of Manitoba
Winnipeg, Manitoba, R3T 2N2, Canada
`rwegner,andersj@cs.umanitoba.ca`

Abstract. In highly complex domains such as disaster rescue, today's autonomous agents simply do not have the ability to perform successfully on their own: the environment is difficult to traverse and to sense accurately, time is a significant factor, and the unpredictable nature of the environment tends to preclude the ability to produce extensive plans for future activity. In this paper we describe an approach to multi-robot control for such environments that focuses on combining the limited abilities of a modern autonomous agent together with human control, in order to produce a teleautonomous system that supports blending the desires of a robot with the wishes of its human controller. We describe the implementation of this approach using simulated Pioneer robots, and evaluate the approach in comparison to autonomous and teleoperated agents in a rescue domain.

1 Introduction

Urban search and rescue (USAR), the exploration of damaged or collapsed urban structures in search of disaster victims, is both a major potential application of AI technology and a current challenge problem for researchers in AI and robotics. USAR is an extremely difficult task for an autonomous agent to perform adequately given the current state of the art in robotic and agent technology. The environment is difficult to maneuver within, and unpredictable in that even a known building layout may have changed dramatically during the associated disaster. Basic robotics skills such as localization are strongly affected (for example, mobile debris causes wheel slippage, leading to more severe errors from wheel encoders), and sensing is much more difficult than any standard indoor domain. The wide range of skills necessary for an agent to perform adequately coupled with the unpredictability of the domain lead most existing efforts to rely heavily on human teleoperation of robotic units (including those whose use at the World Trade Center was widely publicized [1,2]).

This reliance on teleoperation can also be seen in current research. Like other challenge problems such as robotic soccer, USAR research is evaluated in controlled conditions using a physical testbed (e.g. the NIST testbed [3], where the goal is to provide a map to the locations of human victims within an area of debris representing a collapsed structure). While these domains have been described as simplistic compared to real-world USAR [4], the vast majority of entries to such competitions are teleoperated. For example, at both AAAI-02 in Edmonton [5] and IJCAI-03 in Acapulco [6] we were one of only two entries running fully autonomously.

Beyond a desire as AI researchers to advance AI itself, there are good reasons behind a desire to avoid pure teleoperation. Casper and Murphy, for example, describe the operator fatigue that occurs very quickly in real-world rescue situations, and the associated errors in both control and in recognizing visual cues [1, 2]. There are also significant problems with providing situational awareness (that is, a functional mental view of the space within which a robot is operating) to a human operator, and teleoperators also suffer from cognitive overload in terms of processing information [7]. Cognitive overload not only requires information presentation to be very selective, but strongly limits the number of robots that can be controlled by an individual.

We are interested in providing functional intelligent control to a team of robotic rescue agents. Given the difficulty of operating within this domain, the state of the art in autonomous agent technology, and the problems associated with pure teleoperation, a combination of the two approaches (commonly known as a *teleautonomous* approach) is warranted. Ideally, an intelligent control mechanism should support a blend of teleoperation and autonomy that is blended as seamlessly as possible, allowing a teleoperator to focus attention on the problems that require the most assistance. Agents should ideally only interrupt a teleoperator when context suggests it is worth doing so; at the same time, the actions of any agent performing autonomously should be able to be overridden by an operator at any time.

We have developed an approach to blending teleoperation and autonomous control in behaviour-based robotic agents [8]. This consists of three sets of facilities. First, a schema-based [9] autonomous control system for navigation and mapping that allows agents to perform autonomously (subject to the associated limitations that a domain as difficult as this one places). Second, support for direct teleoperation, including a joystick-based interface as well as the ability to control agents at a higher level by setting waypoints. Finally, facilities for blending autonomy and teleoperation appropriately. These consist of a mediation system that allows the blending of the desires of both teleoperator and agent for low-level robotic control, and an intervention recognition system for recognizing both problematic (e.g. agent is stuck) and helpful (e.g. potential victim found) situations in which the operator should be interrupted.

In this paper, we focus on the facilities we have developed for blending teleoperation and autonomy appropriately for robotic rescue agents, including the design and implementation of the software and an evaluation of its performance. Before describing these, we begin with a brief review of related literature.

2 Related Literature

The most well-known early work in combining teleoperation and autonomy is that of Arkin and Ali [7]. Arkin and Ali describe two approaches for teleautonomy with respect to multiple agents. Both of these are schema-based [9] approaches, where behaviors (wander, avoid, etc.) are encoded in the form of motor schemas, which are activated by perceptual schemas (defining perceptual items of interest) and interact at run time to produce output to robot effectors. The first approach has the operator's control (input from a joystick) as a behavior that influences the robots' effectors just as any internal behavior does. The second approach for teleautonomy involves having the operator act as a supervisor. The operator has access to the behavioral parameters of the society (e.g. the low level gains of each motor schema). The operator could effect the emergent behavior of the society of agents as a whole by adjusting their behavioral parameters. This work is limited by its simple blending and effect on an entire group of agents at once, but showed the utility of a teleautonomous approach. Blending an operator's desires as a schema along with the agent's desires was also implemented in the control of a teleautonomous hummer [10], but shares the same limitations of Arkin and Ali's original approach.

Crandall et al. [11] present the notion of *neglect* in remotely controlled agents. They describe neglect as the amount of time during which the agent is not receiving some sort of instruction. They show that this down time can hinder the performance of the robot, and can be due to the operator turning his or her attention away from the agent, or from delays between issuing commands and the agent receiving those commands. They also describe a robot control system consisting of a set of robot behaviors and a user interface for controlling the agents. Their systems use five levels of autonomy ranging from fully autonomous to dormant. However, they do not describe an implementation in their work to show that any balancing has been implemented.

Trividi et al. [12] designed a system that is intended to allow robotic units to recognize traffic collisions and other accidents. This system is strictly a laboratory design and years away from being deployable, but makes use of teleautonomous robotic agents that can form a perimeter around a collision. These agents specialize in forming a perimeter, and the remote operation provides very basic instructions to guide the robots to form perimeters around specific areas. This application of teleautonomy demonstrates the potential to have equipment constantly monitoring an area without the full attention of an operator, but is extremely simplistic: the agents have one purpose, and can achieve that fairly simply through a polygon forming algorithm where each agent takes the role of a point on the polygon. The operator supplies only location guidelines for the polygon forming activity, and the balance between autonomous ability and remote control has been fixed as well.

Murphy and Sprouse [13] describe a strategy for mixing robot and human control in the USAR domain by assigning a different search task to the operator than to an autonomous robot. The robot would perform a systematic search of an area, covering the entire area by splitting the area into sections and applying its senses to each section. The operator then performed the semantic search; in this case the operator directed the robot to semantically similar areas of interest. Murphy et al. [14] describe a paradigm for automating victim detection by robotic agents, while the operators controlled the navigational system. They implement their strategy on a three-agent society architecture, where the robot, human and an Intelligent Assistant Agent together composed the society.

3 Design and Implementation

In our approach to blended teleautonomy, robotic agents are implemented using a schema-based [10] architecture with behaviours suitable for autonomous performance (navigation, mapping, victim identification) in USAR environments. Commands can be accepted from a teleoperator via a joystick facility for low-level direction of a selected individual agent, or via setting high-level waypoints. A mediation component is used to appropriately blend the commands from a teleoperator with agents' autonomous processing, while an intervention recognition component recognizes situations in which an operator should be informed the intervention on his or her part is required. These two components, central to the implementation of blended teleautonomy, are described in the remainder of this section. Peripheral components such as the user interface are described fully in [8].

3.1 Mediator

The mediation component is responsible for smoothly integrating the operator's commands with those of an agent's autonomous control system. While previous approaches have focused on blending operator instructions directly with instructions from an autonomous controller, our approach is more flexible, allowing the agent to intelligently evaluate instructions before blending to ensure that instructions are safe and appropriate to execute. To blend autonomy and teleoperation appropriately, the agent is capable of reasoning about commands that have been sent to it from the human operator. Some commands may be followed to the letter, while others integrated with the agent's own desires or completely refused. The latter allows the veto of actions that would put the robot in danger inadvertently, such as being told to move forward when the operator is not necessarily aware the robot is on the edge of a drop. There may certainly be cases where putting a robot at risk may be deliberate (i.e. the value of information obtained is worth the potential loss of the robot), and so it is also possible for the mediator to allow the operator's commands to be unquestioned.

The mediator operates in one of five modes that are set by a human operator. The most complex of these is *Weighted Teleautonomy*, intended to be the "normal" mode in which agents operate, where the mediator observes the current system, and weights inputs from the teleoperator's interface and the autonomous control system. The user interface provides a slide control allowing a base ratio to be set. This sliding autonomy setting is only one component affecting the weight of autonomy vs. teleoperation, however - commands are also examined in their execution context and weighted based on effect, as will be explained shortly. In contrast to this weighted mode, *Fully Autonomous* and *Fully Teleoperated* modes simply set the weight of one of the two sources to a zero value. In addition to these, we have developed two modes that allow a human operator to have more detailed control. In *Manual Behaviour Weight Modification*, the operator manually defines the internal weights an agent places on its behavours, allowing the operator to alter how the agent runs autonomously, while in *Manual Behaviour Switching*, the operator can switch through the behaviours that are implemented for the autonomous agent, and the agent runs autonomously using only the chosen behaviour. Together, these allow the subsumption of previous approaches to teleautonomy within a single architecture.

A copy of the mediator runs on each agent on a robotic team and actually serves as the main control loop on the robot. It signals the perceptual system to refresh currently perceived information, requests an action vector from the robot's autonomous control system, and if in a mode where an operator is participating, retrieves exterior control signals. These are passed to the *command evaluator*, a symbolic system that is responsible for identifying commands whose execution is dangerous or counter-productive in the current execution context. This is done by predicting the position of the robot if the command were executed, and whether that position would leave the robot in a negative situation from the standpoint of the knowledge in the command evaluator. Once commands are adjusted they are further blended depending on the degree of autonomy set by the operator if the weighted teleautonomy control mode is being used. This results in a control vector that is interpreted and sent to the robot's actuators.

The ideal command evaluator would have enough knowledge to deal with any potential situation in USAR. Given the breadth of this problem, however, complete knowledge is unreasonable to expect. We are currently expanding our knowledge engineering efforts in this area, but have implemented knowledge for two particular situations that are intuitively useful in the USAR domain: moving too near an obstacle (which would potentially get the robot stuck or damaged through contact) and moving away from a potential victim. The evaluation in Section 4 is based on this knowledge.

3.2 Intervention Recognition

Appropriately balancing operator instructions and autonomous abilities is only one part of this overall approach; the other major component is the recognition of situations where operator intervention is required, in order to minimize the cognitive demands on the operator. Recognizing when agents require operator intervention in this approach requires examining specific situations in the form of an intervention recognition system. The intervention recognition system is ultimately responsible for indicating when the balance between autonomy and teleoperation of agents should be changed, by requesting a change of control to the operator (the infrastructure exists in this implementation for requesting assistance from other agents as well, but we have not yet implemented code in the autonomous control mechanism to deal with requests for such assistance). The intervention recognition system performs its task through the use of a knowledge base estimating the degree of likelihood that an agent can or should carry on. The intervention recognition system runs on each individual agent, analyzing its perceptions, identifying specific scenarios indicative of the need for operator intervention, and separating these from the situations where progress is still likely without intervention. The intervention recognition is designed in an extendable manner so that specific scenarios of interest can be encoded in a knowledge-based fashion, resulting in a system that can be used in a wide range of environments.

For the purposes of encoding knowledge useful to the USAR, we have currently implemented three specific scenarios within the intervention recognition system that we have found to be useful. The simplest problem to address, but the most common, is a robot becoming stuck or otherwise immobile. The intervention recognition system identifies when the agent is stuck and signals the operator. A stuck agent is defined as any agent that is sending instructions to its actuators, but the actuators are not completing those instructions. If the agent's actuators are receiving commands, the agent will compare its current sensor readings to past sensor readings attempting to distinguish if there is any evidence supporting movement on the agent's part. If there is little or no evidence supporting movement within the last few perceive-act cycles, the agent is declared stuck.

In addition to becoming stuck, agents can become lost or unable to complete their goals. The agent described here contains no elaborate world model. However, the agent is able to distinguish certain objects (*landmarks*) in the environment uniquely using its sensors, and has limited ability for tracking such objects, which is used by the intervention recognition system to support the identification of a lost or confused agent. The agent remembers how many times it has sensed a landmark and how much time has elapsed since the last time it has sensed the same landmark. The intervention recognition system uses this information to determine when an agent has returned to the same location too often or when an agents has returned to the same location too many times in a specified period of time. In either case, the operator should be notified so that the agent can be encouraged to explore different locations in the environment instead of spending too much time in the same area.

The most crucial event that can occur in the USAR domain, however, is the detection of victims. Victim identification is an extremely difficult task to perform well purely autonomously [4,14], and so is one of the primary reasons why an operator would desire to be interrupted. In our approach, the intervention recognition system is responsible for identifying when an object in the environment resembles a victim and notifying the operator. The intent is for the operator to make a judgment whether a victim is at the location, since the agent is likely to make errors in victim identification. An accurate model of victim identification is not the focus of this work, and for the purposes of evaluation as such vision alone is used to identify objects resembling victims by their color using a single perceptual schema. For future deployment in competition, we intend to supplement this by searching for shapes indicative of partial human forms as well as other sensing such as heat detection.

When the intervention recognition system identifies a situation that requires the operator to intervene, the operator is notified through the user interface (described in more detail in [8]). Briefly, the user interface contains a list of the current available robots and their states. When the intervention recognition system identifies a situation where intervention is desirable, it changes the state of the current robot, updating the user interface. An operator working with the user interface can see that the agent requires assistance, along with a brief message describing the agent's current state, and is able to operate the agent by clicking on the agent's tab on the user interface.

The Intervention Recognition System is implemented in Java as a package containing the *intervention event* objects, the *intervention recognition* object and a *perception memory* object. The perception memory stores snapshots of the agents perceptions for the past five perceptual cycles as an array of perception instances. The perceptions stored in the perceptual memory do not attempt to create a representation of the world other than the basic landmarks described above: they are stored as raw perceptions that the perceptual schemas can use to identify interesting things in the environment.

There are three intervention event objects, polled regularly by the intervention recognition object to inquire if the state of the agent must be changed. These three objects address each of the three important conditions described above: *confused identifier*, *stuck identifier* and *victim identifier*. Each of these event objects contains a link to the current perceptions of the robot via the perception class. The stuck identifier object looks at the agent's current coordinates and compares them to the agent's location four cycles previous. If the current location and the location four cycles ago are the same, and there is a movement instruction being sent to the robot with a speed higher than 0, the agent is considered stuck. The victim identifer relies solely on the victim perceptual schema mentioned previously, while the confused identifier relies on counting the number of times an agent has been within perceptual range of any given landmark (a perceptual schema for landmark identification must be supplied for any domain in which these agents are deployed). If the agent surpasses the threshold, then it is identified as confused. The system is extendible since new intervention events can be coded as additional intervention event objects.

4 Evaluation

In order to examine the performance of this approach, we placed agents in a controlled simulated USAR environment implemented using the Player/Stage simulation tool [15]. Player/Stage was chosen because it is widely used and allows development of code that operates directly on Pioneer robots. Each robot was a Pioneer II equipped with a differential drive, a SICK laser range scanner and a video (CCD) camera with a wide angle lens. The environment used was

$20m^2$, and for the purposes of experimental trials environments were generated and evaluated for equal difficulty and for the purposes of repeatability. Environments were constructed for a desired degree of obstacle coverage (5%, 10%, 15%, and 20%) , using $50cm^2$ obstacles to construct impassable walls with openings between them in order to approximate the structure of a collapsed building. Limits were set on the number of open areas (max. 20) that were generated as a result of this, their size ($100cm^2$-$300cm^2$) and the number of openings to each area (at most 3). Single obstacles were then distributed throughout the environment to make up the desired obstacle coverage. Obstacles were distributed randomly except for two criteria. First, between every obstacle and every open area there was a minimum distance of 120cm, in order that multiple obstacles could not cluster too closely to open areas, thereby reducing the likelihood of creating areas of the environment that completely inaccessible. While inaccessible areas will of course occur in the real world, for the purposes of producing comparable domains we need to control for this. The distance between the center of any two obstacles in the environment could also not be less than 50cm, making it impossible for obstacles to physically overlap more than a few centimeters.

After generating sample environments, the number of local minima present was averaged, and environments were rejected that had local minima counts that were off more than a small range from that mean. Further, environments were also made consistent in difficulty by hand-verifying that there were no inaccessible areas of the domain, and that open areas did not get generated in such a manner that they formed hallways that were too narrow for the robot to physically occupy.

Each environment had 10 victims and 5 negative victims (objects that from a distance appear to be victims). These were distributed randomly except for a proviso that the distance between the center of any real or negative victim from the next closest real or negative victim was at least 60cm.

For the purposes of the simulated environment, the visual schema for victim identification was implemented using colour blob detection, where agents are able to distinguish between actual victims and objects that only resemble victims by their color (negative victims) when they are within $3m$, while outside of $3m$ both victims and objects resembling victims are identified as objects of interest. While victim identification is not the focus of this work, this method serves to force the agent to move within a close proximity in order to make a victim identification, something that would be expected in the real world.

Potential landmarks in the simulated environment were labelled with bar codes that can be read using the laser range finder. While this is not consistent with the real world, the intent here was to allow a consistent and repeatable means of marking landmarks in the world to examine the operation of the confused identifier object.

We examined the performance of the blended teleautonomous approach based on the components described in Section 3 in comparison to purely autonomous and purely teleoperated implementations using the same interface across varying degrees of obstacle coverage. In all cases the same single human teleoperator was used. Extensive results are detailed in [8]; because of space limitations we present a subset of the results related to the coverage of the environment by a team of three agents, the time agents spent immobile, the number of victims found by a team of agents and the number of interactions between the operator and the control system.

Figures 1, 2 and 3 show the performance of the three control systems in terms of area coverage over time for three of the four categories of obstacle coverage (5% coverage was similar enough to 10% coverage to omit, given space limitations). Teleautonomous agents performed significantly better than autonomous agents in terms of area coverage across all degrees of obstacle coverage. We attribute this performance to a human operator's ability to recognize unexplored areas of the environment quickly and guide agents to unexplored areas more efficiently then the autonomous control system could. Some unexplored areas were unlikely to be found by the autonomous agents because of the unique obstacle configurations in those unexplored areas. That is, while we check to ensure the robot can physically fit in any hallways formed, they may still be narrow enough that the robot's motor schema for avoiding obstacles would slow down the agent's progress. The teleoperated agents performed slightly better than the blending agents at the 15 and 20% obstacle coverage levels (though they did not at 5 and 10% obstacle coverage levels), since although the operator could guide blending agents into unexplored areas, once an agent was neglected (i.e. the operator shifted attention to another agent) the autonomous portion of the blending control system could guide the robot back to an explored area. This happened less at lower obstacle coverage levels: since there are fewer obstacles, there are fewer course changes necessary for agents to go around them when operating without benefit of an operator, and less likelihood of heading back toward an area that an operator just steered the agent away from.

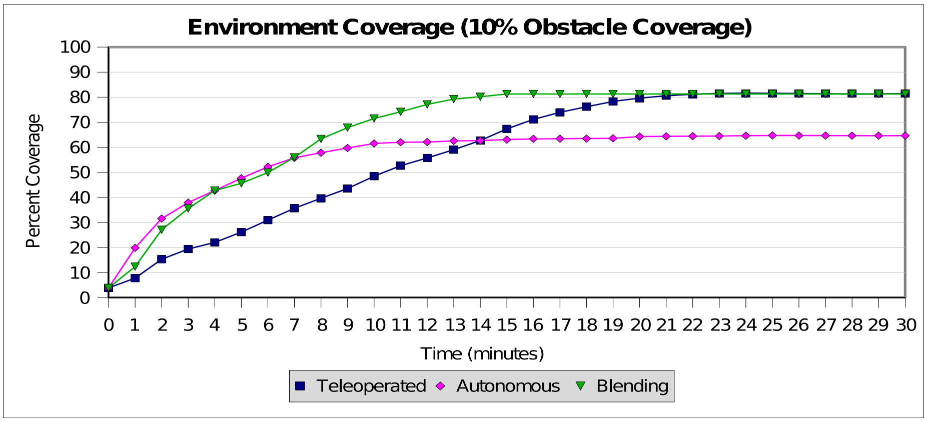


Fig. 1. Average (n=5) environment coverage achieved by autonomous, blended teleautonomous and teleoperated agents in environments where 10% was covered in obstacles.

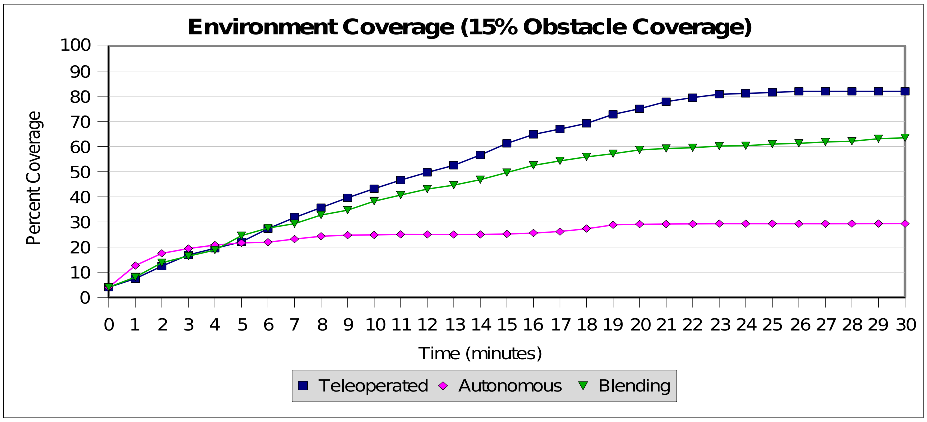


Fig. 2. Average (n=5) environment coverage achieved by autonomous, blended teleautonomous and teleoperated agents in environments where 15% was covered in obstacles.

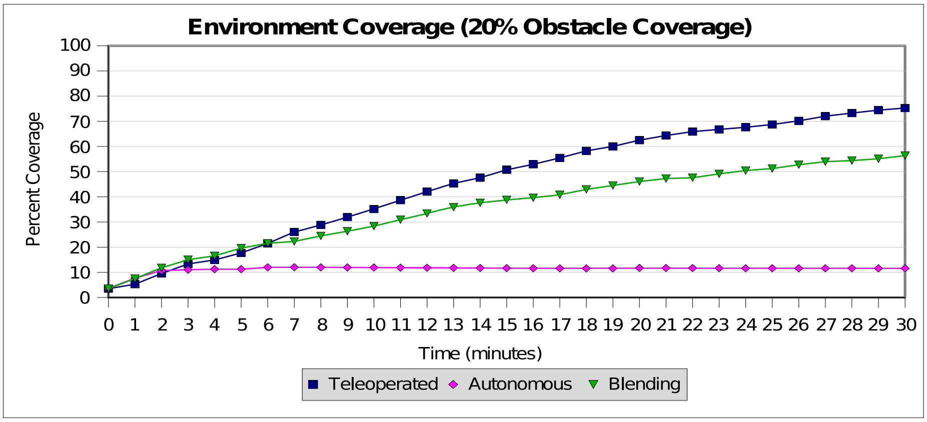


Fig. 3. Average (n=5) environment coverage achieved by autonomous, blended teleautonomous and teleoperated agents in environments where 20% was covered in obstacles.

The time each agent spent immobile with respect to autonomous versus blending agents (see Fig. 4) is another indication of the gains associated with blending autonomy and teleoperation. Since the autonomous agents are behavior-based, they are susceptible to local minima, often becoming stuck in difficult environments. When agents got stuck in autonomous trials, they would often remain stuck. In the blending trials, if an agent became stuck, the operator was often able to free the agent. Since the operator was notified by the intervention recognition system whenever an agent became stuck, the operator was often able to free the agent in a timely manner, reducing the amount of time any particular blending agent spent immobile. In the lower obstacle coverage trials (5% and 10% obstacle coverage), agents became stuck less overall. Moreover, when agents did get stuck, they tended to get stuck less severely, and therefore it was easy for the operator to get the agent mobile again. In trials with higher obstacle coverage, the agents would get themselves stuck in much more complex ways, making it more difficult for operators to release them. In trials where the obstacle coverage was 20%, the time spent stuck for the blending control system was much higher, since agents were often difficult to get mobile, leading to agents being abandoned. Blending operator instructions with the autonomous instructions contributes to a significant increase in effectiveness of agents, which can be observed by comparing the results of the autonomous trials and the blending trials.

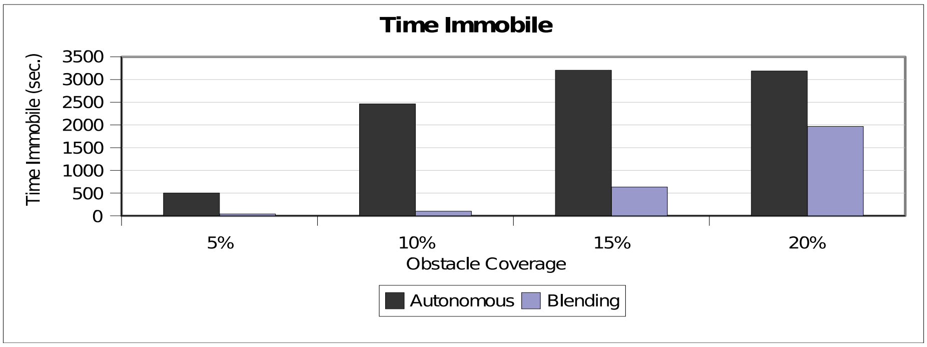


Fig. 4. Average time in seconds spent immobile by environment difficulty, for blending and autonomous agents.

With respect to successful victim identification, we found that the agents using blended teleautonomy had an advantage over both teleoperated agents and autonomous agents (see Figs. 5, 6 and 7). At least a few victims in any experimental scenario were reasonably out in the open and easy enough to navigate to and find autonomously, and both blending agents and autonomous agents could take advantage of this. Correspondingly, only very little attention on the part of the operator was needed for blending agents. Later on in the trials, when the victims in the open were all located, the blending agents performed better then the autonomous agents, because the operator could guide the agents through the more difficult areas of the environment, encouraging the agents to cover more area and discover more victims.

While there were cases (at the 15% and 20% obstacle coverage levels) where purely teleoperated agents could still outperform teleautonomous agents, there is a significant overhead being paid for this in terms of operator intervention and ultimately operator fatigue. Throughout all trials performed, the teleoperated agents required many more interactions to complete their task. This ranged from

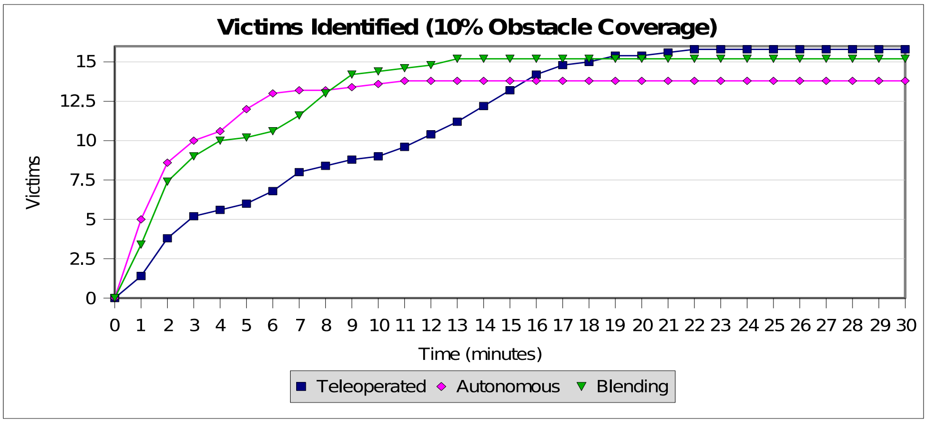


Fig. 5. Comparison of number of victims identified in teleoperated, autonomous, and blending experiments in environments where 10% was covered in obstacles. All results are averages over 5 trials.

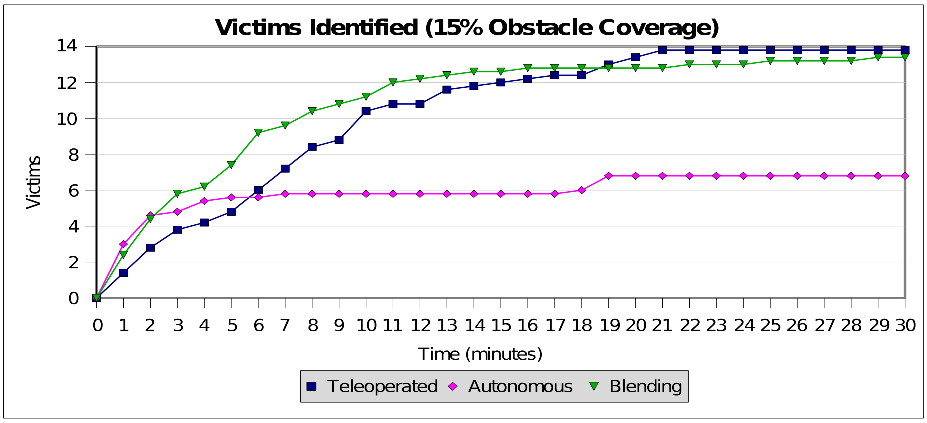


Fig. 6. Comparison of number of victims identified in teleoperated, autonomous, and blending experiments in environments where 15% was covered in obstacles. All results are averages over 5 trials.

an average of 5.9 times more interactions than the blended control system for 5% obstacle coverage, to 1.4 times the number of interactions for 20% obstacle coverage (see Fig. 8). Even with the additional attention required by the more dense environments, the blending control system required less attention from the operator, which contributes to a lower cognitive load.

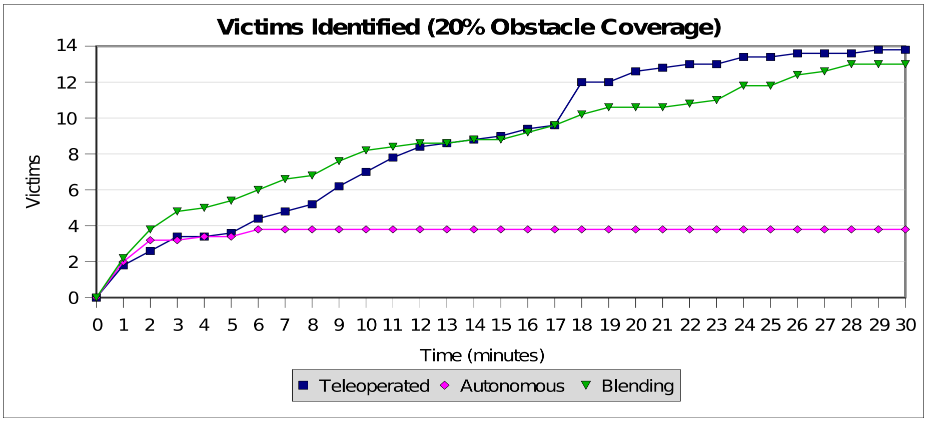


Fig. 7. Comparison of number of victims identified in teleoperated, autonomous, and blending experiments in environments where 20% was covered in obstacles. All results are averages over 5 trials.

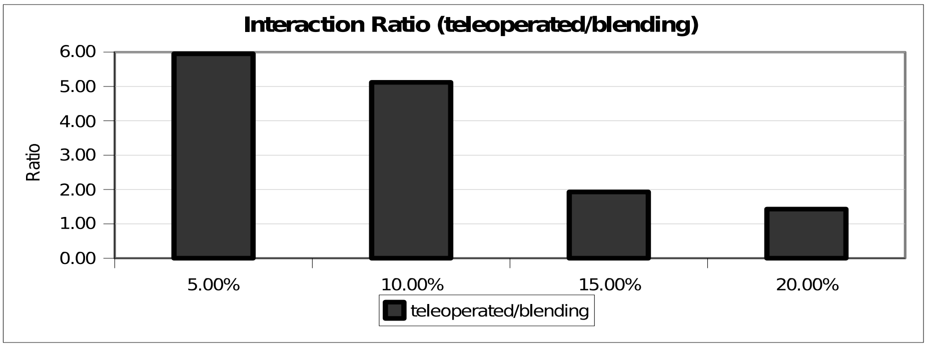


Fig. 8. Average (n=5) ratio of operator interactions (teleoperated/blended teleautonomous in environments with 5, 10, 15, and 20% obstacle coverage.

5 Future Work

There are a number of directions that future work in this area can profitably take. In terms of immediate further experimentation, we are currently examining the efficacy of this approach as the number of robots increases, both to examine how far the benefits obtained can be pushed, and to help identify future improvements that will allow the operator to better control larger teams of agents. We are also working on extending the knowledge bases used by the mediator and the command evaluator, since the broader the range of situations these can recognize and deal with, the better the performance of this approach should be.

One of the most obvious extensions to this work is the application of the blending control system on physical robots. Since this work was done using the Player/Stage application suite, all code written to control the simulated stage agents is directly compatible with physical Pioneer mobile robot platforms. However, the code described in this paper was not verified on a set of physical robots. Extending the blending control system to work with other mobile robot platforms is another goal of future work in this area. There are several issues that have to be addressed if this system is going to be applied to physical robots. First, on real robots, perfect localization is no longer a simple assumption. Odometry on real robots is likely to have at least some noise, and that noise will be cumulative. The application of vision and other sensor technology would have to be employed in order to have useful localization. Another assumption that has to be dealt with is the increase in sensor noise and environment complexity. Vision in particular will be a challenging problem in a real robot compared to a simulated one, and a more sophisticated method for handling errors will have to be developed. The approach and much of the code written for the simulated blending system will be applicable on the physical robots, but the underlying infrastructure will require much additional work.

In terms of the overall approach, the major planned extension of this work is to allow support from peer agents as well as human operators. The potential for this is already present in this approach in that the intervention recognition system has the ability to send requests for advice or assistance to other agents. The mediator was also designed so that it could be extended to include instructions from peer agents instead of human operators. We intend to extend the autonomous control mechanisms to deal with requests for assistance or information from other agents, and integrate other agents into the knowledge bases used by the intervention recognition system and command evaluator. This will allow research into the efficacy of making agents more proactive in terms of assisting one another in this domain (even volunteering information rather than being asked, for example).

6 Conclusion

This paper has described facilities for balancing autonomy and teleoperation effectively for a complex environment, Urban Search and Rescue. Further explanation of the other components that make up a complete teleautonomous navigation system (e.g. the user interface, the waypoint manager, etc.) may be found in [8].

The experiments described in this paper demonstrate that this approach, through blending autonomy and teleoperation appropriately and notifying an operator when intervention is desirable, can significantly improve agent effectiveness. It is also evident that the blending of autonomy and teleoperation reduces the number of interactions between the operator and the agent while still maintaining a comparable level of performance. The results have shown the blending supported by our control mechanism allows a human operator to more accurately control a group of robots, and the system to find victims faster and with fewer errors, than relying on autonomy or teleoperation exclusively.

We have also indicated a number of directions for future experimentation and research.

References

1. Casper, J.: Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. Master's thesis, University of South Florida (2002) Computer Science and Engineering.
2. Casper, J., Murphy, R.: Workflow study on human-robot interaction in USAR. In: Proceedings of the International Conference on Robotics and Automation. Volume 2., Washington (2002) 1997–2003
3. Jacoff, A., Messina, E., Evans, J.: Experiences in deploying test arenas for autonomous mobile robots. In: Performance Metrics for Intelligent Systems Workshop, Mexico City, Mexico (2001)
4. Murphy, R., Casper, J., Micire, M., Hyams, J.: Assessment of the NIST standard test bed for urban search and rescue. In: Proceedings of AAAI Mobile Robotics Competition Workshop, Austin, TX (2000) 11–16
5. Baltes, J., Anderson, J.: A pragmatic approach to robotic rescue: The keystone fire brigade. In: Proceedings of the AAAI-02 Mobile Robot Competition Workshop, Edmonton, Alberta (2002) 38–43
6. Baltes, J., Anderson, J.: The keystone fire brigade 2003. In: Proceedings of the IJCAI-03 Mobile Robot Competition Workshop, Acapulco, Mexico (2003) 30–35
7. Arkin, R.C., Ali, K.S.: Integration of reactive and telerobotic control in multiagent robotic systems. In: Proceedings of the Third International Conference on Simulation of Adaptive Behavior, Brighton, England (1994) 473–478
8. Wegner, R.: Balancing robotic teleoperation and autonomy in a complex dynamic environment. Master's thesis, University of Manitoba (2003) Department of Computer Science.
9. Arkin, R.C.: Behavior-Based Robotics. Cambridge: MIT Press, Cambridge, MA (1998)
10. Arkin, R.C., Balch, T.: Cooperative multiagent robotic systems. In Kortenkamp, D., Murphy, R., Bonasso, R., eds.: Artificial Intelligence and Mobile Robots. Cambridge: MIT Press, Atlanta, Georgia (1998) 278–296
11. Crandall, J.W., Goodrich, M.A.: Experiments in adjustable autonomy. In: Proceedings of the International Conference of Systems, Man, and Cybernetics. Volume 3. (2001) 1624–1629
12. Trivedi, M., Hall, B., Kogut, G., Roche, S.: Web-based teleautonomy and telepresence. In: Proceedings of the 45th Optical Science and Technology Conference. Volume 4120., San Diego (2000) 81–85
13. Murphy, R., Sprouse, J.: Strategies for searching an area with semi-autonomous mobile robots. In: Proceedings of Robotics for Challenging Environments, Albuquerque, NM (1996) 15–21
14. Casper, J., Murphy, R., Micire, M., Hyams, J.: Mixed-initiative control of multiple heterogeneous robots for urban search and rescue. Technical report, University of South Florida (2000)
15. Gerkey, B.P., Vaughan, R.T., Stoy, K., Howard, A., Sukhatme, G.S., Mataric, M.J.: Most valuable player: A robot device server for distributed control. In: Proceedings of Intelligent Robots and Systems, Wailea, Hawaii (2001) 1226–1231

Emotional Pathfinding

Toby Donaldson, Andrew Park, and I-Ling Lin

School of Interactive Arts and Technology,
Simon Fraser University, Burnaby, B.C., Canada
{tjd, aparkd, ilina}@sfu.ca

Abstract. This paper presents a study of the influence of emotions on the behaviour of an intelligent pathfinding agent. A model of pathfinding is proposed that takes into account the emotional state of the agent. Results show that blindly following the most urgent emotion can lead to degenerate behaviour, and that cross-exclusion can be used to effectively moderate emotional influences. Applications of this work include any situation where realistically behaving emotional characters are needed, such as in video games or movies with computer-generated characters.

1 Introduction

The clash between emotion and reason, the head and the heart, is a common theme in our culture. For most of its short history, AI has focused almost exclusively on reason. John McCarthy, for instance, has suggested that while emotions are a real phenomenon distinct from reasoning, their potential to subvert rational behaviour can cause agents to act in confusing and potentially dangerous ways. [1] In McCarthy's mathematical eyes, emotions are essentially noise that ruin an agent's ability to concentrate . In contrast, Marvin Minsky [2] has argued that emotions are one of the physiological tools that intelligent agents can use to great effect, in the same way that people can use fanciful imagining to solve real problems.[1] Over the last decade, AI research into emotion has been spurred by advances in neurology that indicate emotions play a critical role in rational decision-making, perception, and human interaction [3]. Neurologist Antonio Damasio has persuasively argued that the ability to make appropriate decisions in dynamic, complex, and unpredictable environments, depends critically on emotions [4]. Goleman also claims that "emotional intelligence" is more important than traditional notions of intelligence [5]. In humans, intelligence and emotions are inextricably linked, and so if we want to design agents that behave like real people, then we must take emotions seriously.

Emotions have two main applications in AI: modeling and understanding human emotions, and creating lifelike characters that express emotions, e.g. [6, 7,8]. Agents that *recognize* human emotions could have practical applications,

[1] This dichotomy of opinion is reminiscent of the classic *Star Trek* TV series, where Dr McCoy (= Minsky) emphasized emotions and feeling, while Mr Spock (= McCarthy) relied solely on emotionless reason.

e.g. an agent that realizes a user is ecstatic, or despondent, or angry would better be able to interpret and communicate with them (e.g. [9,10]). Most examples of intelligent human-computer interaction are implicitly situated in calm, business-like settings where user frustration is the only emotion that might arise. However, more complex scenarios occur all the time in real life. Consider child custody negotiations between divorcing parents, or life and death scenarios in a hospital emergency room. In these sorts of emotionally charged situations, an intelligent agent would need to take into account the emotions of the people involved, and also deport itself in a sensitive manner. However, actually recognizing emotions in other people can be extremely difficult; even people make mistakes. Not everyone wears their heart on their sleeve, and even for those who do, getting a computer to recognize the gestures, facial expressions, and other physical manifestations of emotions (e.g. tears, red cheeks, frowns, etc.) is no small feat. Thus, we will not consider recognizing human emotions any further in this paper.

The other major application of emotions has been in creating realistic characters for story-telling or game-play. Characters that express realistic emotions are usuallly more interesting to readers/players than emotionless ones. Plus, emotions are important storytelling tools, e.g. we interpret things differently depending upon the emotions we sense in characters. For instance, imagine someone in a dark room. If they are smiling and expectant, then we might be lead to believe that the situation is innocent; perhaps a game of hide and seek. But if the character looks fearful and is trembling, we are likely to believe that something more sinister is going on.

Our interest in this paper is the second application: how can we use emotions to make more interesting characters for stories and, in particular, games? Ultimately, our work could be useful as AI middleware in games. AI middleware is software that provides AI functions (such as genetic algorithms and state machines) to games or simulations, [11]. To narrow down the problem, we focus on how emotions can influence an agent's movements. Pathfinding is of particular interest, because it can be used to model many different situations, and intelligent pathfinding is used in many video games. Plus, in movies with computer generated characters, the characters will typically be moving around, requiring some sort of pathfinding and collision-avoidance.

After discussion theories of emotions and the basics of AI-style pathfinding, we will present a variation of pathfinding that takes into account the emotional state of the agent. The purpose is to create interestingly plausible movements, and understanding what parameters can be used to control and generate them. Finally, we outline a number of possible ideas for future work, such as full-figure modeling.

2 Theories of Emotion

What are emotions? While the term *emotion* has a well-understood popular meaning, Picard [3] reports that there is no agreement among emotion researchers on what counts as an emotion. Typically, emotions are the things that we name by words such as *love*, *hope*, *anger*, *sadness*, *disgust*, *fear*, *joy*, *happiness*, *surprise*, *remorse*, and so on. Human emotions have been well-studied in psychology. Psychological definitions of human emotion typically have the following characteristics [12]:

1. Emotions are *episodic* in nature, meaning they are of limited duration, and gradually fade away.
2. Emotions are *triggered* by an external or internal event.
3. Whenever an emotion is triggered it changes the state of the agent substantially, e.g. it gets aroused, excited, and probably distracted from its current undertaking. Three components, known as the *reaction triad*, are necessary for a state change to be called emotion:
   a) physiological arousal;
   b) motor expression;
   c) subjective feeling.
4. Emotions are *relevance detectors*: they arise when stimuli with important *meaning* for the agent have been detected. This definition helps distinguishes emotions from other similar concepts such as mood, interpersonal stance, attitudes, and personal traits.

Also, emotions are not usually things that people can intentionally turn off and on. Emotions are reactions to stimuli. People can certainly plan to act in a way that will cause or prevent certain emotions, e.g. looking at a picture of a loved one might bring joy. There are a number of psychological theories of emotion. We will not give an exhaustive list of such theories here, instead we will highlight two important aspects of emotional theories: the direction of causality, and continuity versus discreteness.

One basic issue is whether emotions influence physical reactions, or vice versa. For instance, the James-Lange theory [13] says that when a person perceives something, such as a frightening animal, they react with physical (neurovegetative) manifestations. As a consequence of such an unpleasant physical reaction, the person develops fear. On the other hand, the Cannon-Bard theory [14] claims that the frightening stimulus leads, first, to the feeling of fear, which then brings about the physical response. The Cannon-Bard theory has support from experiments that show after removal of all visceral and spinal feedback, cats and dogs can still express emotions — cognitive appraisal of the situation is the main emotional elicitor. Schacter's theory [15] adds a cognitive element to the James-Lange theory, saying that emotions are only felt if bodily arousal exists. Damasio also supports the James-Lange theory with his famous card game (somatic marker hypothesis) [4]. However, in making a computation model of emotional agents, Cannon-Bard theory is more favorable because it has a natural computational interpretation (e.g. [8]).

Theories of emotion can also be distinguished according to whether they model emotions continuously or discretely [12]. A discrete theory of emotion identifies emotion as discrete states, often labeling them with conventional emotional terms. For instance, Plutchik [16] distinguishes between eight basic emotions: fear, anger, sorrow, joy, disgust, acceptance, anticipation, and surprise. In a continuous theory, emotions are points a continuous space, where the axes are continuous scales such as arousal, valence, and control [3].

2.1 The OCC Theory

Another theory of emotions, popular among some AI researchers, is the so-called OCC theory by Ortony, Clore, and Collins [17,18]. In this model, emotions are influenced by events, agents, and objects. Event-related emotions are resentment, hope, fear, joy, distress, satisfaction, disappointment, relief, and so on. Emotions for objects are liking and disliking. Agent emotions include pride, shame, reproach, and admiration. It assumes that emotions are valenced reactions; the intensity of an affective reactions determines whether or not it will be experienced as emotions. Processing an emotion consists of four steps: classifying input stimulus, quantifying the intensity of emotions, mapping intensities to emotional expressions, and, finally, realizing those emotional expressions. One of the reasons that this model is popular among artificial intelligence community is that the processing stages can be implemented in a straightforward way using rules [19].

Due to this computational friendliness, the model of emotional pathfinding we use in this paper will be a simplified version of the OCC model. Our goal in this paper is not to implement a general-purpose emotional architecture, but instead to find ways to make emotion and intelligence interact in interesting and useful ways for games and stories. The ultimate judge of the quality of our algorithm will be players (and readers): if it something *seems* real to a player, then it does not matter if the effect was created in an inhuman way, or if the implementation does not religiously follow the theory. Our goal is not to validate OCC as a theory of emotion; it is merely a practical tool. Another minor liberty we will take with the OCC model is that we will consider *hunger* and *thirst* as if they were emotions. This is partly out of convenience, and the fact that hunger and thirst have properties very similar to emotion. For instance, hunger *increases* over time if the agent does nothing to sate it. This is analogous to how emotions *decay* over time if nothing is done to stoke them. Hunger, like emotions, are involuntary — an agent cannot *directly* cause itself to be in a hungry state, although it can certainly act in a way that is likely to lead to that state. We could replace hunger and thirst, with, say, with needful emotions such as love, amusement, empathy, surprise, etc.

3 Intelligent Pathfinding

Pathfinding is a well-studied problem in AI [20], and the A^* algorithm has proven to be the algorithm of choice in many video games.[2] Briefly, A^* is a graph-search algorithm that finds a path between a given $START$ node and $GOAL$ node. It

[2] The industry website `www.gamasutra.com` is an excellent source of practical information on video games.

works by maintaining a list of all the nodes whose children have not yet been visited, and at every step it chooses from this list the node S with the smallest $g+h$, value, where $g(S)$ is the (true) distance from $START$ to S, and $h(S)$ is the estimated heuristic distance from S to $GOAL$. If the heuristic function h never *over*estimates the true distance to $GOAL$), then we say that h is *admissible*, and A^* is then guaranteed to find a shortest path. Unfortunately, this optimality comes at the cost of tremendous amounts of memory: the list of candidate nodes can be exponentially large, and so in practice this severely limits the number of nodes that can be explored.

If we are willing to live with potentially sub-optimal solutions, then numerous other algorithms can be used. For instance, *hill-climbing* algorithms do not store any candidate nodes, and instead immediately step to the next node with the best h, or $g+h$ value. Such *greedy* approaches are well-known for getting stuck in local extrema, and so some technique for avoiding or getting out of local extrema is necessary. For instance, after hill-climbing makes a certain number of steps, a *random re-start* could be made, which starts the hill-climbing all over again, saving the best path as it goes. A popular variation on hill-climbing is *simulated annealing*, where the next state is chosen probabilistically with the probability based on a constantly falling temperature value, and the scores of the candidate nodes. Another popular approach is to use a *population* of candidate solutions, and to apply various transformations to these candidates, often inspired by natural processes. For instance, evolutionary algorithms, genetic algorithms, ant systems, and swarm algorithms all maintain a population of candidate states, and modifying them in various ways to achieve a new population [21,22]. These methods often have a large number of parameters that must be tuned through trial and error, or even using other search methods. For certain problems, these methods turn out to be extremely effective strategies, although in practice it is usually necessary to add as much domain-specific information as possible to make the algorithm effective.

3.1 Pathfinding in Video Games

We are interested in the A^* algorithm in large part because it is commonly used in video games to control the movements of characters. For instance, imagine a 2D game where the player's character is a knight in shining armor, and the background map is the player's kingdom. To move the knight, the player clicks with a mouse on the spot on the map where they want to go, and the knight automatically ambles to that spot, intelligently navigating around obstacles along the way. Typically, the map is abstracted into a planar grid of squares (or sometimes hexagons, e.g. [23]), and a graph is constructed from the grid by putting a node in each square that the knight is allowed to be in. Two nodes/squares are considered connected if the knight can move directly between them without impediment. Obstacles are thus represented by the absence of nodes, or node connections. Since the map is on a 2D-plane, the heuristic function h is normally the straight-line Euclidean distance between the current position of the knight and the clicked-point; this guarantees that h is admissible, since, on a plane, the shortest distance between two points is always a straight line. The location of the knight is the $START$ node, and the node the user clicks in becomes the $GOAL$ node. Immediately after the user selects the $GOAL$, A^* is run to find a best path, and when its done, the knight begins walking along the path. If an obstacle moves into its path after it is has finished searching, then it will need to re-plan its path.

As mentioned, A^* search can eat up significant amounts of time and memory, and so in real-time applications (like games) sometimes an answer must be given before the algorithm has finished. In such cases, the knight may need to make moves before it knows the best path to follow. Sometimes, it will make wrong turns early on that will take it down a wrong path, resulting in U-turns or odd routes that look anything but intelligent. If the player always expects the agent to take the optimal route between two points, then non-optimal routes may be perceived as glitches, which can often detract from the game.[3] However, in games that aim for realism, players can accept non-optimal routes if there is some good reason why they are non-optimal. For instance, if there's a battle going on, then, understandably, the knight may be highly stressed, and might not always make optimal decisions. Or if the character is grieving over the loss of a friend, it may move more slowly, or less intelligently. The upshot is that for games, or simulations, that aim for realistic models of human-like agent behaviour, then purely rational and intelligent behaviour is *not* as realistic as behaviour that makes sub-optimal decisions in a natural-seeming way. That's because humans are not always rational, and for emotional and other reasons, sometimes make what appear to be irrational decisions.

Good story-telling requires that characters have emotions, and display and act upon these emotions in ways that the reader/player can sensibly interpret. In movies and big-budget video games, lead characters are often based on real human actors (e.g. using "motion capture"), or an animator is put in charge of carefully crafting all their movements. Using trained actors or hand-animating a graphical character is relatively expensive, and so it is often a limiting factor. Our goal is to develop software that can be used to help design autonomous agents that behave in emotionally realistic ways, but at a fraction of the cost of using human actors or animators. One practical application is the automation of crowd scenes, where hand-animating numerous minor characters would be too tedious or resource-consuming. It's easy to imagine movie-like scenarios where, say, an explosion causes a large crowd of people to react in a highly emotional way. Instead of scripting the behavior of each and every person, we could use emotionally-sensitive software to have the agents react in realistic and interesting ways. We will say more about such possibilities in the Future Work section at the end of the paper.

[3] Some glitches can cause unintended amusement. For example, in the original *Sims* game, you could trap characters in rooms by strategically (or even accidentally) placing empty dinner plates on the floor. The characters refused to step over them, and so would eventually die unless you moved the plates.

4 Combining Emotions and Pathfinding

In our work, we have combined A^* search and a variation of the OCC model of emotions. We have decided to initially concentrate on modeling a problem described by Minsky in [2]. In Minsky's problem, the agent is driven by thirst and hunger. Unfortunately, all the food is at the North end of the map, and all the water is far away at the South end. Suppose, for the sake of explanation, that the agent has hunger = 10, and thirst = 9. Since its hunger is more urgent than its thirst, the agent goes North to the food, and by eating some food its hunger becomes, say, 8. Now it is more thirsty than hungry, so it treks all the way South to the water, where it quenches it thirst by 2 points; now hunger = 8, and thirst = 7. Now it is more hungry than thirsty, so it turns around to go North for some more food. The problem is clear: the agent will move back and forth between the food and the water in response to its emotional flip-flopping. Indeed, it is not hard to concoct scenarios where the agent goes back and forth indefinitely (until it dies), since hunger and thirst naturally increase over time: if the distance the agent travels from the food, to the water, and back to the food is long enough, it's hunger could increase by more than the 2 points one bite of food is worth.

Our approach to solving this problem is the one suggested by Minsky: *cross-exclusion*. Cross-exclusion is a very general idea that can be applied to any group of related units, and it essentially causes the agent to listen to only one unit at a time. In our scenario, there are just two relevant units, *hunger* and *thirst*. The idea is that when *hunger* is satisfied (by eating food), it becomes positively excited. This positive excitement automatically causes a negative *inhibitory* signal to be sent to *thirst*. Thus *hunger* remains as the most urgent need, and so the agent stays at the food, eating until it is no longer hungry. When *hunger* is satisfied, it stops inhibiting *thirst*, causing the agent to go South for water. The same thing happens with *thirst*: when *thirst* is excited, *hunger* is proportionally inhibited. The effect is that the agent does the right thing in this example, which is to completely fulfill one need before attending to the other.

For cross-exclusion to work, the inhibitory values must be set high enough to ensure that there is no flip-flopping of states after one need is satisfied. We determined these values through trial and error, although it would be possible to automate this process using, for instance, an evolutionary algorithm. Among humans, different people have different thresholds for hunger, thirst, and so on, thus there is no one best set of threshold values. Indeed, since the ultimate goal is to create agents that do interesting things in the context of games or movies, we want to keep the ability to experiment with unusual parameter values. In some cases, the flip-flopping behaviour we see when cross-exclusion is turned off may be just what a story calls for.

4.1 Pseudocode for Emotional Pathfinding

In our experiments we use a single agent whose main search procedure is called Emotional-Search (see Figure 1). It takes four pieces of input: a starting node, a set of goals to be achieved, the emotional state E, and the A^* heuristic function h. The agent is either MOVING, THINKING, or PANICKING. When it is THINKING, we consider it to be consciously performing a variation of A^* search we call *emotional* A^*, or eA^* for short. When the agent is MOVING, it takes *speed* number of steps along the *Path* return by the last call to eA^*. When it is in the PANIC state, it makes random moves. After each step and each call to eA^*, the agent re-calculates its emotional state E, and its *speed*.

```
EMOTIONAL-SEARCH(Start-node, Goalset, E, h)
curr-node ← Start-node
status ← THINKING
while Goalset ≠ ∅
      do E ← CALC-EMOTIONAL-STATE(Goalset, curr-goal, curr-node, E)
         speed ← CALC-NEW-SPEED(speed, Goalset, curr-goal, curr-node, E)
         Goalset ← Goalset ⋃ GENERATE-NEW-GOALS(Goalset, curr-goal, curr-node, E)
         if status = PANICKING
           then curr-node ← random neighbor of curr-node
                if E['fear'].intensity < PANIC-THRESHOLD
                  then status ← THINKING
         elseif status = MOVING
           then curr-node ← next node along Path
                if curr-node = curr-goal
                  then Goalset ← Goalset − curr-goal
                       status ← THINKING
                if Path has no more nodes
                  then status ← THINKING
         elseif status = THINKING
           then curr-goal ← select (but don't remove) most urgent goal of Goalset
                Path ← eA*(Goalset, curr-goal, curr-node, E, h)
                Path ← first speed nodes of Path
                status ← MOVING
         if E['fear'].intensity > PANIC-THRESHOLD
           then status ← PANICKING
```

```
GENERATE-NEW-GOALS(Goalset, curr-goal, curr-node, E)
thirsty ← E['thirst'].intensity > E['thirst'].threshold
hungry ← E['hunger'].intensity > E['hunger'].threshold
goals ← ∅
if thirsty and no WATER goal in Goalset
   then goals ← goals ⋃ {new WATER goal}
if hungry and no FOOD goal in Goalset
   then goals ← goals ⋃ {new FOOD goal}
return goals
```

```
eA*(Goalset, curr-goal, E, h)
h_e ← CALC-h_e(Goalset, curr-goal, curr-node, E, h)
Path ← A*(curr-goal, curr-node, h_e)
return Path
```

Fig. 1. This code was implemented in Python, which has proven to be a good language for rapid prototyping of AI-type applications.

The emotional state E is treated as a mapping between emotion names and emotion data structures. Following the OCC model, our emotions essentially consist of three numeric values: D=desirability, P=potentiality, and I=intensity. For our current experiments, potentiality and intensity are the most relevant values. Intensity is the value that applications will look at when they want to query the emotional state. Potentiality is the value that is updated when some stimulus affects an agent, e.g. when it eats food or drinks water. Intensity is calculated as function of potentiality. Storing two values like this allows for greater flexibility in how we deal with changing emotions. For instance, cross-exclusion requires that some emotions inhibit others, essentially masking the true intensity of other emotions. We can handle this in a neat way by keeping the true value of the emotion as potentiality, performing the inhibiting in the function that generates the intensity. Thus, when the inhibiting emotion finally ceases, the true value of the other emotions, which have been backed up as potentialities, can be easily and naturally re-established. Emotions have a number of other values, such as an update function that defines how the emotion decays over time; hot and cold threshold values that mark extreme values for that emotion. For example, if *hunger* is above its hot threshold, then the agent will add a "satisfy hunger" goal to *Goalset*.

The PANIC emotion overrides any other emotion. The idea is that extreme fear is such an important survival technique for people that it ultimately contributes to survivability: agents that panic in the face of danger are more likely to run away to panic another day. Speed is also affected by PANIC, e.g. panicking agents usually move (a lot) more quickly than calm ones. It is possible that some agents reaction to PANIC may be to curl up in a ball and not move at all. PANIC-SPEED could be treated as a personality parameter, although it might not be a very interesting behaviour for a lead character. However, in a crowd of panicking characters, if a few characters react in unexpected ways then the scene may be more interesting.

The agent's *speed* depends upon the level of its fear, hunger, and thirst. Essentially, if the agent is very fearful, it moves most quickly, and if it is very hungry or thirsty it moves more slowly. The eA^* algorithm creates a modified version of h (the heuristic function) called h_e. It then calls ordinary A^* search using h_e as the heuristic function. The h_e functions we used in the experiments for this paper are linear combinations of the h function, where the coefficients are chosen as functions of hunger, thirst, and fear. For instance if the agent is very hungry, thirsty, or frightened, then h_e is a "noisy" variation of h, resulting in less intelligent moves. If the agent is slightly less hungry, thirsty, or fearful, then h_e is a slightly less noisy version of h. This means that h_e is usually inadmissible, and so in a highly emotional state the agent will be distracted from its pathfinding and will likely find less than optimal routes.

GENERATE-NEW-GOALS is responsible for adding new goals to the agents *Goalset*. Essentially, if the agent becomes hungry or thirsty, and it does not already have a hunger or thirst goal, one is added.

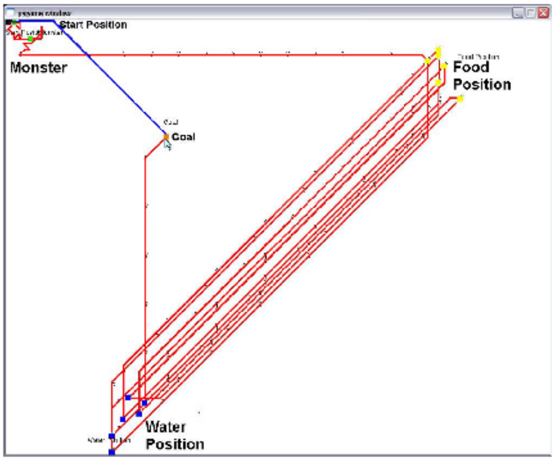


Fig. 2. An Agent that always follows its most urgent emotion. The food is in the north-east part of the map, and the water is in the south-west part. The agent starts in the upper-left corner, and ends where the arrow is point near the start point. The questions marks on the path indicate when the agent is thinking, i.e. running eA^*. Note the zig-zags near the starting node: the agent has bumped into a monster.

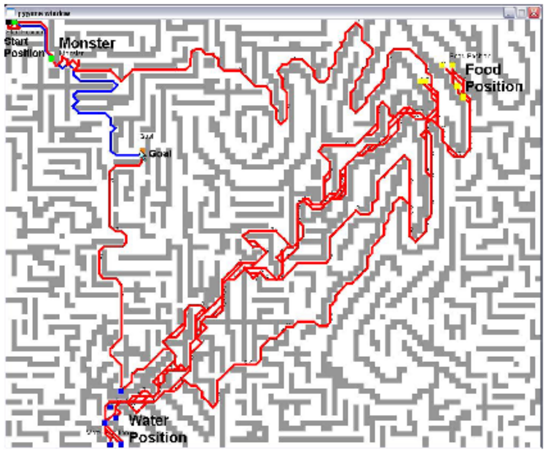


Fig. 3. An Agent that always follows its most urgent emotion, in a maze. The same as Figure 2, but this time within a maze.

5 Results

We show some of our results in the accompanying four figures. The first two show the agent's behaviour without cross-exclusion, and with and without a maze. The second two show the change in behaviour when we add cross-exclusion. In both

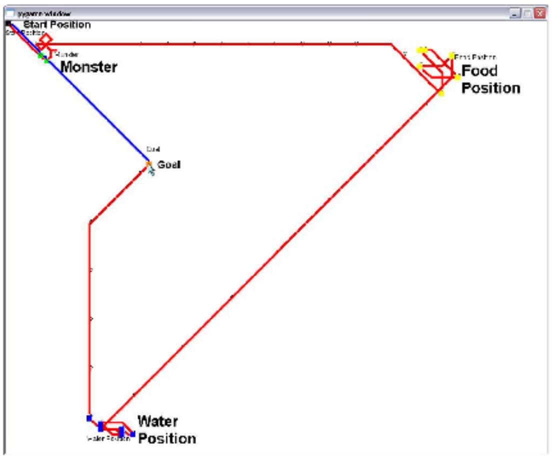


Fig. 4. More intelligent pathfinding using cross-exclusion. The same as Figure 2, but this time cross-exclusion is used. The result is that the agent completely satisfies its most pressing need before moving on to other ones.

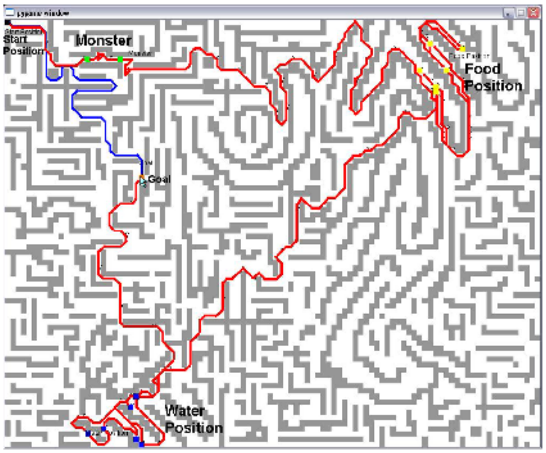


Fig. 5. More intelligent pathfinding using cross-exclusion, with a maze. The same as Figure 4, but this time within a maze.

cases, there are "monsters" near the start of the maze that cause the agent to become very fearful. Thus, sometimes you will see small tangles in the path where the agent panicked. It eventually calms down (the monsters stay put!) and returns to more thoughtful pathfinding.

In Figure 2 and Figure 3, we see an example of the back-and-forth behaviour of an agent that always follows its most urgent emotion. Figure 4 and Figure 5 show the path when the agent's emotions are cross-excluded. Clearly, the cross-exclusion paths are more intelligent-looking, because they visit food and water in a sensible order, and are much shorter. In a game, this is almost certainly the sort of path a computer controlled character should follow. The yo-yo behaviour that we see without cross-exclusion is certainly interesting, because it very clearly points out the problem of always following your most urgent emotion.

6 Future Work

The work presented in this paper represents the beginning of a larger project involving the design of realistic agents for games or movies. We plan to further refine the main simulation loop, and also streamline the implementation of emotions. While the OCC is popular, it is relatively complex, and we want to make a small number of parameters that a designer can modify to get a desired behaviour. We also plan to make the various behaviours of the agent configurable through a graphical interface.

As we have shown in this paper, the emotional state of an agent can significantly affect how it moves, but demonstrating this only by movement in a maze severely limits how an agent can show its emotions. Thus, we are currently investigating platforms for full-body animation, where the agent's entire body can be used to express emotions. For instance, [24] uses emotional transforms to convert ordinary human movement into the same movement but with an emotional flavor, such as happy or angry. That work is concerned only with the movements of the figure, and does not consider how emotions affect and agent's decision-making. Furthermore, we would like to model scenarios with multiple agents, where who they talk to, where they go, and what they do is realistically influenced by their emotional state. For instance, [25] reports on a start on this problem, and it shows that interesting and realistic emergent behaviours arise in groups when emotion-like influences are taken into account.

Acknowledgments. Thank you to Bonnie Cheng, Tom Calvert, and anonymous reviewers for a number of helpful suggestions and improvements.

References

1. McCarthy, J.: Making robots conscious of their mental states. `http://www-formal.stanford.edu/jmc/consciousness/consciousness.html` (1995)
2. Minsky, M.: The Society of Mind. Simon and Schuster (1986)
3. Picard, R.: Affective computing. Technical Report 321, MIT Media Laboratory, Perceptual Computing Section (1995)
4. Damasio, A.: Descartes' Error: Emotion, Reason, and the Human Brain. Avon (1995)
5. Goleman, D.: Emotional Intelligence : Why It Can Matter More Than IQ. Bantam (1997)

6. Bates, J.: The role of emotion in believable agents. Communications of the ACM **37** (1997) 122–125
7. Velásquez, J.: When robots weep: Emotional memories and decision-making. In: Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98), AAAI Press (1998)
8. Gratch, J., Marsella, S.: Tears and fears: modeling emotions and emotional behaviors in synthetic agents. In Müller, J.P., Andre, E., Sen, S., Frasson, C., eds.: Proceedings of the Fifth International Conference on Autonomous Agents, Montreal, Canada, ACM Press (2001) 278–285
9. APE-2001: 2nd workshop on attitude, personality and emotions in user-adapted interaction, at International Conference on User Modelling, UM2001 (2001)
10. Marsella, S.C.: Pedagogical soap. In Dautenhahn, K., Bond, A.H., namero, L.C., Edmonds, B., eds.: Socially Intelligent Agents Creating Relationships with Computers and Robots. Kluwer (2000)
11. Dybsand, E.: Ai middleware: Getting into character part 1–5. Gamasutra (`www.gamasutra.com`) (2003)
12. Thiele, A.: Lecture note 4 on cognition and emotion, `http://www.staff.ncl.ac.uk/alex.thiele/` (2003)
13. James, W.: The Principles of Psychology. Harvard University Press (1890)
14. Cannon, W.B.: The jameslange theory of emotion: A critical examination and an alternative theory. American Journal of Psychology **39** (1927) 10–124
15. Papanicolaou, A.C.: Emotion: A Reconsideration of Somatic Theory. Routledge (1989)
16. Plutchik, R.: A general psychoevolutionary theory of emotion. In: Theories of Emotion. Volume 1. Academic Press (1980)
17. Ortony, A., Clore, G.L., Collins, A.: The Cognitive Structure of Emotions. Cambridge University Press (1988)
18. Bartneck, C.: Integrating the occ model of emotions in embodied characters. In: Workshop on Virtual and Conversational Characters: Applications, Methods, and Research Challenges. (2002)
19. Adamatti, D., Bazzan, A.: A framework for simulation of agents with emotions. In: WORKCOMP 2002 (Workshop de Computacao). (2002)
20. Pearl, J.: Heuristics – Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley (1984)
21. Michalewicz, Z., Fogel, D.B.: How to Solve It: Modern Heuristics. Springer-Verlag (1999)
22. Kennedy, J., Eberhart, R.: Swarm Intelligence. Morgan Kaufmann (2001)
23. Yap, P.: Grid-based pathfinding. In Cohen, R., Spencer, B., eds.: Advances in Artificial Intelligence: 15th Conference of the Canadian Society for Computational Studies of Intelligence, AI 2002. Volume 2338/2002 of Lecture Notes in Computer Science., Springer-Verlag (2002) 44
24. Amaya, K., Bruderlin, A., Calvert, T.: Emotion from motion. In: Graphics Interface. (1996) 222–229
25. Calvert, T., Sonnichsen, A.: Emergent behaviour in animated crowd scenes. In: Life Like Computer Characters Workshop. (1998) Snowbird, Utah.

Combining Evidence in Cognate Identification

Grzegorz Kondrak

Department of Computing Science,
University of Alberta,
Edmonton, AB, T6G 2E8, Canada
`kondrak@cs.ualberta.ca`
`http://www.cs.ualberta.ca/~kondrak`

Abstract. Cognates are words of the same origin that belong to distinct languages. The problem of automatic identification of cognates arises in language reconstruction and bitext-related tasks. The evidence of cognation may come from various information sources, such as phonetic similarity, semantic similarity, and recurrent sound correspondences. I discuss ways of defining the measures of the various types of similarity and propose a method of combining then into an integrated cognate identification program. The new method requires no manual parameter tuning and performs well when tested on the Indoeuropean and Algonquian lexical data.

1 Introduction

Cognates are words of the same origin that belong to distinct languages. For example, French *lait*, Spanish *leche*, and Italian *latte* constitute a set of cognates, since they all derive from Latin *lactem*. In general, the number of cognates between related languages decreases with time, and the ones that remain become less similar. Recurrent sound correspondences, which are produced by regular sound changes, are helpful in distinguishing cognate pairs from accidental resemblances. For example, the fact that /d/:/t/ is a recurrent correspondence between Latin and English (*ten/decem*, *tooth/dentem*. etc.) indicates that Latin *die* 'day' is not cognate with English *day*.

Depending on the kind of data, the task of cognate identification can be defined on three levels of specificity:

1. Given a pair of words, such as English *snow* and German *schnee*, compute the likelihood that they are cognate.
2. Given a list of word pairs matched by meanings, such as the one in Table 1, rank the pairs according to the likelihood that they are cognate.
3. Given a pair of vocabulary lists, such as the one in Table 2, produce a ranked list of candidate cognate pairs.

A phonetic measure can be computed for any pair of words in isolation (levels 1, 2 and 3), but a longer list of related words is necessary for the determination of

Table 1. An excerpt from the German/Albanian word-pair list [10].

1.	‘all’	alə	ɟiθə
2.	‘and’	unt	e
3.	‘animal’	tīr	kafšə
4.	‘ashes’	ašə	hi
5.	‘at’	an	nə
6.	‘back’	rükən	špinə
7.	‘bad’	šlext	kec
8.	‘bark’	rində	škəlbozə
9.	‘because’	vayl	sepse
10.	‘belly’	bawx	bark

the recurrent sound correspondences (levels 2 and 3), while a semantic measure is only applicable when words are accompanied by glosses (level 3).

The ultimate goal of the research described in this paper is the fascinating possibility of performing an automatic reconstruction of proto-languages from the information contained in the descendant languages. Given dictionaries of related languages, a hypothetical language reconstruction program would be able to determine recurrent sound correspondences, identify cognate sets, and reconstruct their corresponding proto-forms.

The identification of cognates is not only the key issue in language reconstruction, but is also important in a number of bitext-related tasks, such as sentence alignment [3,19,21,24], inducing translation lexicons [11,18], and improving statistical machine translation models [1]. Most of the applications take advantage of the fact that nearly all co-occurring cognates in bitexts are mutual translations. In the context of bitexts, the term *cognate* usually denotes words in different languages that are similar in form and meaning, without making a distinction between borrowed and genetically related words.

Current approaches to cognate identification employ either phonetic/orthographic similarity measures [2,19,21,23] or recurrent sound/letter correspondences [6,18,26]. However, there have been very few attempts to combine different ways of cognate identification. Yarowsky and Wincentowski [28] bootstrap the values of edit cost matrix with rough phonetic approximations, and

Table 2. Excerpts from the Cree (left) and the Ojibwa (right) vocabulary lists [9].

āniskōhōčikan	string of beads	*āšikan*	dock, bridge
asikan	sock, stocking	*anaka’ēkkw*	bark
kamāmakos	butterfly	*kipaskosikan*	medicine
kostāčīwin	terror, fear	*kottāčīwin*	fear, alarm
misiyēw	large partridge, hen	*mēmīkwan’*	butterfly
namēhpin	wild ginger	*mississē*	turkey
napakihtak	board	*namēpin*	sucker
tēhtēw	green toad	*napakissakw*	plank
wayakēskw	bark	*tēntē*	very big toad

then iteratively re-estimate the matrix in order to derive empirically observed character-to-character probabilities. Kondrak [13] linearly combines a phonetic score with a semantic score of gloss similarity.

In this paper, I present a method of integrating distinct types of evidence for the purpose of cognate identification. In particular, the combined phonetic and correspondence-based similarity measures are applied to lists of word pairs, and the semantic similarity of glosses is added on when dealing with vocabulary lists. The new method combines various similarity scores in a principled way. In terms of accuracy, when tested on independently compiled word and vocabulary lists, it matches or surpasses the results obtained using the method with manually set parameters [13]. Finally, the method makes it possible to utilize complex, multi-phoneme correspondences for cognate identification.

The paper is organized as follows. The next three sections provide background on the measures of phonetic, correspondence-based, and semantic similarity, respectively, in the context of cognate identification. After introducing the method of combining various measures, I describe and discuss experimental results on authentic language data. I conclude with a comparison of the method presented here with another method of identifying cognates.

2 Phonetic Similarity

Surface-form similarity of words can be estimated using orthographic and/or phonetic measures. Simple measures of orthographic similarity include edit distance [19], Dice's bigram similarity coefficient [2], and the Longest Common Subsequence Ratio (LCSR) [21]. Phonetic measures are applicable if words are given in a phonetic or phonemic transcription. ALINE [12] is a phonetic word aligner based on multivalued phonetic features with salience weights. Thanks to its ability to assess the similarity of individual segments, ALINE performs better on cognate identification than the orthographic measures that employ a binary identity function on the level of character comparison [13].

ALINE returns a normalized score in the $[0, 1]$ range. The score by itself can be used to rank candidate pairs with respect to their phonetic similarity. However, in order to combine the phonetic score with the semantic and/or correspondence-based scores, it is helpful to convert the score assigned to a pair of words into the probability that they are related. For modeling the distribution of scores, I adopt the Beta distribution. The Beta distribution is defined over the domain $[0, 1]$, and has two free parameters $\hat{A}$ and $\hat{B}$. The relationship between the two parameters and the mean and variance of the distribution is the following:

$$\mu = \frac{\hat{A}}{\hat{A} + \hat{B}} \qquad\qquad \sigma^2 = \frac{\hat{A}\hat{B}}{(\hat{A} + \hat{B})^2(\hat{A} + \hat{B} + 1)}$$

Figure 1 shows the distribution of phonetic scores between word pairs in the development set (Cree–Ojibwa) within the 0.04 intervals. The left and right

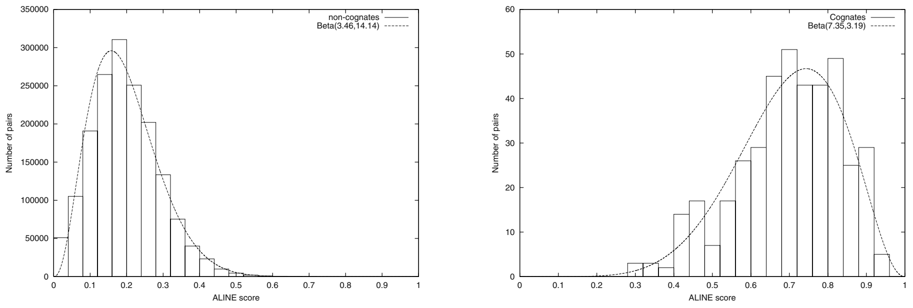


Fig. 1. Distribution of the phonetic scores for the unrelated (left) and the cognate (right) word pairs, and the corresponding Beta distributions.

plot depict the phonetic score distribution for the unrelated and for the cognate word pairs, respectively. The parameters of the corresponding Beta distributions were calculated from the mean and variance of the scores using the relationship expressed in formulas (1) and (2). For unrelated words, the Beta distribution fits the distribution of phonetic scores remarkably well. For cognate words, the fit is also quite good although somewhat less tight, which is not surprising considering that the number of cognate pairs is several magnitudes times smaller than the number of unrelated pairs.

3 Correspondence-Based Similarity

3.1 Determination of Simple Correspondences

For the determination of recurrent sound correspondences (often referred to simply as correspondences) I employ the method of inducing a *translation model* between phonemes in two wordlists [14]. The idea is to relate recurrent sound correspondences in wordlists to translational equivalences in bitexts. The translation model is induced by combining the maximum similarity alignment with the competitive linking algorithm of Melamed [22]. Melamed's approach is based on the *one-to-one* assumption, which implies that every word in the bitext is aligned with at most one word on the other side of the bitext. In the context of the bilingual wordlists, the correspondences determined under the *one-to-one* assumption are restricted to link single phonemes to single phonemes. Nevertheless, the method is powerful enough to determine valid correspondences in wordlists in which the fraction of cognate pairs is well below 50% [14].

The correspondence-based similarity score between two words is computed in the following way. Each valid correspondence is counted as a link and contributes a constant positive score (no crossing links are allowed). Each unlinked segment, with the exception of the segments beyond the rightmost link, is assigned a smaller negative score. The alignment with the highest score is found using dynamic programming [27]. If more than one best alignment exists, links are assigned the weight averaged over the entire set of best alignments. Finally, the score is normalized by dividing it by the average of the lengths of the two words.

3.2 Determination of Complex Correspondences

Kondrak [15] proposed an extension of the one-to-one method that is capable of discovering complex, ‘many-to-many” correspondences. The method is an adaptation of Melamed’s algorithm for discovering non-compositional compounds in bitexts [20]. A non-compositional compound (NCC) is a word sequence, such as “high school”, whose meaning cannot be synthesized from the meaning of its components. Experimental results indicate that the method can achieve up to 90% recall and precision in determination of correspondences on vocabulary lists [15].

When the NCC approach is applied, the computation of the similarity score is slightly modified. Segments that represent valid NCCs are fused into single segments before the optimal alignment is established. The contribution of a valid correspondence is weighted by the length of the correspondence. For example, a correspondence that links three segments on one side with two segments on the other side is given the weight of 2.5. As before, the score is normalized by dividing it by the average of the lengths of the two words. Therefore, the score for two words in which all segments participate in links is still guaranteed to be 1.0.

Figure 2 shows the distribution of correspondence-based scores between word pairs in the development set (Cree–Ojibwa). For unrelated words, the fit with the Beta distribution is not as good as in the case of phonetic scores, but still acceptable. For cognate words, the Beta distribution fails to account for a number of word pairs that are perfectly covered by correspondences (score = 1.0). However, the problem is likely to be less acute for language pairs that are not as closely related as Cree and Ojibwa.

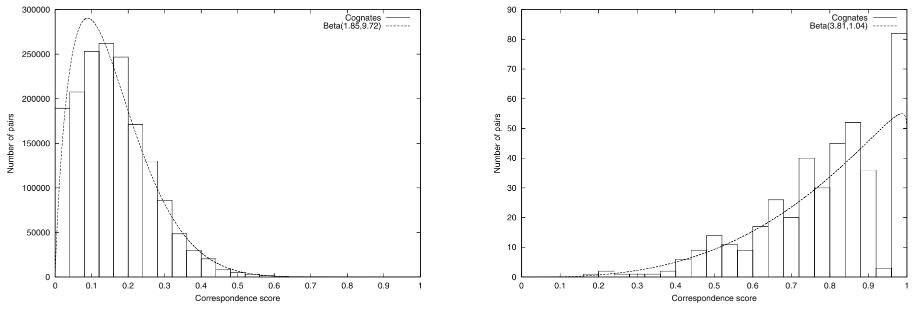


Fig. 2. Distribution of the correspondence-based scores for the unrelated (left) and the cognate (right) word pairs, and the corresponding Beta distributions.

3.3 Determination of Correspondences in Vocabulary Lists

Kondrak [14] showed that correspondences determined with the one-to-one approach can be successfully used for cognate identification in pairs of word lists, where the words are already matched by meanings. However, the task is more challenging when cognates have to be identified in unstructured vocabulary lists, In vocabulary lists, as opposed to word lists, words across languages are not neatly matched by meanings; rather, semantic similarity has to be inferred from glosses. Whereas in word lists of related languages the percentage of cognates can be expected to exceed 10%, the probability that randomly selected words from two vocabulary lists are cognate is usually less than 0.1%. Attempting to determine correspondences with such a small signal-to-noise ratio is bound to fail. It is necessary first to identify a smaller set of likely cognate pairs, on which a translation model can be successfully induced.

One possible way to determine the set of likely cognate pairs is to select n candidate pairs starting from the top of the ordered list produced by a combined semantic and phonetic approach. However, the selected pairs are likely to include many pairs that exhibit high phonetic similarity. When a translation model is induced on such set, the strongest correspondences can be expected to consist mostly of pairs of identical phonemes.

A better approach, which is not biased by the phonetic similarities between phonemes, is to select candidate pairs solely on the basis of semantic similarity. The idea is to extract all vocabulary entries characterized by the highest level of semantic similarity, that is, gloss identity. Even though such a set is still likely to contain mostly unrelated word pairs, the fraction of cognates may be sufficiently large to determine the strongest correspondences. The determined correspondences can then be used to identify cognates among all possible word pairs.

4 Semantic Similarity Features

Kondrak [13] developed a scheme for computing semantic similarity of glosses on the basis of keyword selection and WordNet [5] lexical relations. The scheme combines four lexical relations and two focus levels, which together yield eight semantic similarity levels (Table 3). Keywords are salient words extracted from glosses by a heuristic method based on part-of-speech tags. If there exists a lexical relationship in WordNet linking the two glosses or any of their keywords, the semantic similarity score is determined according to the scheme shown in Table 3. The levels of similarity are considered in descending score order, with keyword identity taking precedence over gloss hypernymy. The scores are not cumulative. The numerical values in Table 3 were set manually on the basis of experiments with the development set (Cree–Ojibwa).

In this paper, I propose to consider the eight semantic similarity levels as binary semantic *features*. Although the features are definitely not independent, it may be advantageous to consider their combinations rather than just simply

Table 3. Semantic similarity features and their numerical scores [13].

Lexical relation	Focus level	
	Gloss	Keyword
Identity	1.00	0.50
Synonymy	0.70	0.35
Hypernymy	0.50	0.25
Meronymy	0.10	0.05

$$\begin{tikzcd}
\text{gloss identity} \arrow[dr] \arrow[dd] & \\
& \text{keyword identity} \arrow[dd] \\
\text{gloss synonymy} \arrow[dr] \arrow[dd] & \\
& \text{keyword synonymy} \arrow[dd] \\
\text{gloss hypernymy} \arrow[dr] \arrow[dd] & \\
& \text{keyword hypernymy} \arrow[dd] \\
\text{gloss meronymy} \arrow[dr] & \\
& \text{keyword meronymy}
\end{tikzcd}$$

Fig. 3. Partial ordering of semantic features.

the most prominent one. For example, gloss hypernymy accompanied by keyword synonymy might constitute stronger evidence for a semantic relationship than gloss hypernymy alone.

In the context of detecting semantic similarity of glosses, a transitive *subsumption* relation can be defined for the semantic features. In the following assertions, the expression "feature A subsumes feature B" should be understood as "feature B is redundant if feature A is present".

1. Gloss identity subsumes other relations involving glosses (e.g. gloss identity subsumes gloss meronymy).
2. Keyword identity subsumes other relations involving keywords.
3. Features involving a lexical relation between glosses subsume features involving the same lexical relation between keywords (e.g. gloss hypernymy subsumes keyword hypernymy).
4. Synonymy subsumes hypernymy and meronymy, and hypernymy subsumes meronymy.

The resulting partial ordering of features is shown in Figure 3. Assertion 4 is probably the most debatable.

I investigated the following variants of semantic feature ordering:

LN The linear order that corresponds to the semantic scale originally proposed in [13].

Table 4. Symbols used in Section 5.

sem	=	related to semantic similarity
ph	=	related to phonetic similarity
rc	=	related to correspondence-based similarity
$+$	=	related to cognate pairs
$-$	=	related to unrelated pairs
$cogn$	=	the given pair of words are cognate
$\neg cogn$	=	the given pair of words are unrelated
v	=	feature vector for the given word pair
v_j	=	values of the binary semantic features
s	=	numerical scores for the given word pair (in the $[0,1]$ range)
σ	=	partial similarity scores
α	=	interpolation parameters (weights)
d	=	probability density function of the corresponding Beta distribution
C_i	=	normalizing constants independent of the given word pair

SP The partial order implied by assertions 1–4, which is shown in Figure 3.
WP The weaker version of the partial order, implied by assertions 1–3.
NO The unordered set of the eight semantic features.
MK The feature set corresponding to Method K in [13], which contains only the WordNet-independent features: gloss identity and keyword identity (the former subsumes the latter).
MG The feature set corresponding to Method G in [13], which contains only gloss identity.
NS Empty set, i.e. no semantic features are used (the baseline method).

I discuss the effect of the feature ordering variants on the overall accuracy in Section 6.

5 Combining Various Types of Evidence

In [13], the overall similarity score was computed using a linear combination of the semantic and the phonetic scores. The interpolation parameter was determined on a development set. In this paper, I adopt the Naive Bayes approach to combining various sources of information. The vector v consists of the eight semantic features, the phonetic similarity score, and the correspondence-based similarity score. The overall word-pair similarity score for a pair of words is computed by the following formula (see Table 4 for the explanation of symbols):

$$\begin{aligned} score &= \frac{p(cogn \mid v)}{p(\neg cogn \mid v)} = \frac{p(cogn) \cdot p(v \mid cogn)}{p(\neg cogn) \cdot p(v \mid \neg cogn))} = C_1 \cdot \frac{p(v \mid cogn)}{p(v \mid \neg cogn))} \\ &= C_2 \cdot \left(\frac{(\prod_j p(v_j \mid cogn))}{(\prod_j p(v_j \mid \neg cogn))} \right)^{\alpha_{sem}} \cdot \left(\frac{d_{ph+}(s_{ph})}{d_{ph-}(s_{ph})} \right)^{\alpha_{ph}} \cdot \left(\frac{d_{rc+}(s_{rc})}{d_{rc-}(s_{rc})} \right)^{\alpha_{rc}} \end{aligned}$$

It is more convenient to do the computations using logarithms:

$$\log score = \alpha_{sem} \cdot \sigma_{sem} + \alpha_{ph} \cdot \sigma_{ph} + \alpha_{rc} \cdot \sigma_{rc} + C_3$$

where

$$\sigma_{sem} = \sum\nolimits_j \log \frac{p(v_j \mid cogn)}{p(v_j \mid \neg cogn)}, \quad \sigma_{ph} = \log \frac{d_{ph+}(s_{ph})}{d_{ph-}(s_{ph}))}, \quad \sigma_{rc} = \log \frac{d_{rc+}(s_{rc})}{d_{rc-}(s_{rc})}.$$

Since the goal is a relative ranking of candidate word-pairs, the exact value of the normalizing constant C_3 is irrelevant.

The difference between the current method of combining partial scores and the method presented in [13] lies in the way the original scores are transformed into probabilities using the Naive Bayes assumption and the Beta distribution. A number of parameters must still be established on a separate training set: the conditional probability distributions of the semantic features, the parameters for the beta distributions, and the interpolation parameters. However, the values of the parameters (except the interpolation parameters) are set automatically rather than manually.

6 Results

The cognate identification methods were tested on two different data sets: a set of structured word lists of 200 basic meanings, and a set of unstructured vocabulary lists containing thousands of entries with glosses.

The accuracy of the methods was evaluated by computing the 11-point interpolated average precision for the vocabulary lists, and the n-point average precision for the word lists (n is the total number of cognate pairs in a list). The output of the system is a list of suspected cognate pairs sorted by their similarity scores. Typically, true cognates are very frequent near the top of the list, and become less frequent towards the bottom. The threshold value that determines the cut-off depends on the intended application, the degree of relatedness between languages, and the particular method used. Rather than reporting precision and recall values for an arbitrarily selected threshold, precision is computed at a number of different recall levels, and then averaged to yield a single number. In the case of the 11-point average precision, the recall levels are set at 0%, 10%, 20%, ..., 100%. In the case of the n-point average precision, precision is calculated at each point in the list where a true cognate pair is found. In the experiments reported below, I uniformly assumed the precision value at 0% recall to be 1, and the precision value at 100% recall to be 0.

6.1 Results on the Indoeuropean Word Lists

The experiments in this section were performed using a list of 200 basic meanings that are considered universal and relatively resistant to lexical replacement [25]. The development set included six 200-word lists (Italian, Polish, Romanian, Russian, Serbocroatian and Spanish) adapted from the Comparative Indoeuropean

Table 5. The average cognate identification precision on the Indoeuropean 200-word lists for various methods.

Languages		Phonetic	Simple	Complex	Phonetic + Simple	Phonetic + Complex
English	German	.916	.949	.924	.946	.930
French	Latin	.863	.869	.874	.881	.882
English	Latin	.725	.857	.740	.828	.796
German	Latin	.706	.856	.795	.839	.830
English	French	.615	.557	.556	.692	.678
French	German	.504	.525	.526	.575	.572
Albanian	Latin	.618	.613	.621	.696	.659
Albanian	French	.612	.443	.460	.600	.603
Albanian	German	.323	.307	.307	.395	.398
Albanian	English	.277	.202	.243	.340	.330
Average		**.616**	**.618**	**.605**	**.679**	**.668**

Data Corpus [4]. The test set consisted of five lists (Albanian, English, French, German, and Latin) compiled by Kessler [10]. In this experiment, only words belonging to the same semantic slot were considered as possible cognates.

Table 5 compares the average cognate identification precision on the test set obtained using the following methods:

Phonetic. The phonetic approach, in which cognate pairs are ordered according to their phonetic similarity score computed by ALINE. The settings of ALINE's parameters are the same as in [12].

Simple. The correspondence-based approach, as described in [14] (method D), in which only simple, one-to-one correspondences are identified.

Complex. The correspondence-based approach that identifies complex, many-to-many correspondences [15].

Phonetic + Simple. The combination of the phonetic and the correspondence-based approaches, without utilizing complex correspondences.

Phonetic + Complex. The combination of the phonetic and the correspondence-based approaches that utilizes complex correspondences.

In the final two variants, the phonetic and the correspondence-based approaches are combined using the method described in Section 5, with the parameters derived from the Italian/Polish word list (the interpolation parameters α_{ph} and α_{rc} were held equal to 1). This particular language pair was chosen because it produced the best overall results on the development set. However, the relative differences in average precision with different training sets did not exceed 1%.

The results in Table 5 show that both the phonetic method and the correspondence-based method obtain similar average cognate identification precision. The combination of the two methods achieves a significantly higher precision. Surprisingly, the incorporation of complex correspondences has a slightly negative effect on the results. A close examination of the results indicates that few useful complex correspondences were identified by the NCC algorithm in the

Table 6. The average precision on the Algonquian vocabulary lists obtained by combining the semantic similarity features, the phonetic similarity score, and the complex-correspondence-based similarity score.

Languages		NS	MG	MK	LN	SP	WP	NO
Fox	Menomini	.488	.607	.640	.651	.652	.652	.491
Fox	Cree	.508	.682	.678	.698	.694	.694	.549
Fox	Ojibwa	.655	.674	.685	.691	.695	.695	.572
Menomini	Cree	.438	.591	.612	.618	.613	.608	.523
Menomini	Ojibwa	.478	.611	.632	.641	.639	.635	.516
Average on test set		**.513**	**.633**	**.649**	**.660**	**.658**	**.657**	**.530**
Cree	Ojibwa	.717	.783	.785	.787	.784	.784	.722

200-word Indoeuropean lists. This may be caused by the small overall number of cognate pairs (57 per language pair, on average) or simply by the paucity of recurrent complex correspondences.

Additional experiments showed that straightforward averaging of the phonetic and the correspondence-based scores produces results that are quite similar to the results obtained using the method described in Section 5. On the test set, the straightforward method achieves the average precision of .683 with simple correspondences, and .680 with complex correspondences.

6.2 Results on the Algonquian Vocabulary Lists

The cognate identification method was also tested on noun portions of four Algonquian vocabulary lists [9]. The lists representing Fox, Menomini, Cree, and Ojibwa contain over 4000 noun entries in total. The results were evaluated against an electronic version of the Algonquian etymological dictionary [8]. The dictionary contains 4,068 cognate sets, including 853 marked as nouns. The Cree–Ojibwa language pair was used as the development set, while the remaining five pairs served as the test set. The proportion of cognates in the set of word-pairs that have at least one gloss in common was 33.1% in the development set and ranged from 17.5% to 26.3% in the test set.

Table 6 shows the average precision obtained on the Algonquian data by combining the phonetic, semantic and (complex) correspondence-based similarity using the method presented in Section 5. The columns correspond to variants of semantic feature ordering defined in Section 4. The numbers shown in bold type in the left-most four columns can be directly compared to the corresponding results obtained using the method described in [13]. The latter method, which uses the linear combination of the phonetic and the semantic similarity scores (set according to Table 3), achieved the 11-point average precision of .430, .596, .617, and .628, for variants NS, MG, MK, and LN, respectively. Therefore, the improvement ranges from 5% (LN) to nearly 20% (NS). When all semantic features are utilized (columns LN, SP, WP, and NO), there is hardly any difference in average precision between alternative orderings of semantic features

Table 7. The average precision on the Algonquian vocabulary lists (test set only) obtained by combining various methods.

Methods	Correspondences		
	None	Simple	Complex
—	—	.448	.473
Phonetic	.430	.472	.513
Semantic	.227	.633	.625
Phonetic + Semantic	.631	.652	.660

(LN, SP, WP). However, applying the features without any ordering (NO) is almost equivalent to using no semantics at all (NS).

Table 7 provides more details on the contribution of various types of evidence to the overall average precision. For example, the merger of the phonetic and the semantic similarity with no recourse to correspondences achieves the average precision of .631. (not significantly better than the average precision of .628 obtained using the method described in [13]). Replacing the phonetic similarity with the (simple) correspondence-based similarity has little influence on the average precision: .448 vs .430 without semantics, and .633 vs. .631 with semantics. The advantage provided by complex correspondences all but disappears when all types of evidence are combined (.660 vs. .652). Relying on gloss similarity alone is inadequate (.227) because no continuous score is available to order candidate pairs within the semantic similarity classes.

The tests were performed with the following parameter settings: semantic feature ordering – linear (LN); parameters for computing phonetic similarity – as in [13]; parameters for computing the correspondence-based score – as in [14] (complex correspondences limited to consonant clusters); number of iterations of the NCC algorithm — 12, as in [15]. When two types of evidence were combined, the interpolation parameters were held equal to 1. With all three types of evidence, the interpolation parameters were $\alpha_{sem} = 2$, $\alpha_{ph} = 1$, and $\alpha_{rc} = 1$.

The choice of values for the interpolation parameters requires further explanation. The weights used for the final testing were selected because they are relatively simple and result in near-maximum average precision on the training data. They also have a theoretical justification. Both the phonetic and the correspondence-based similarity measures are calculated on the basis of the phonetic transcription of words. Moreover, recurrent correspondences are composed mostly of similar or identical phonemes. In contrast, the semantic similarity measure is based exclusively on glosses. The experiments performed on the training set suggested that the best results are obtained by assigning approximately equal weight to the gloss-based evidence and to the lexeme-based measures. The results in Tables 7 and 6 reflect this observation. If the weights are equal for all three types of evidence, the average precision drops to .616 with the simple correspondences, and to .639 with the complex correspondences.

7 Computing Similarity vs. Generating Proto Projections

It is interesting to compare the method described here to the method that was originally used to compile the etymological dictionary [8], which served as our gold standard, from the vocabulary lists [9], which also constituted our test data in Section 6.2. The method [7] is based on generating proto-projections (candidate proto-forms) of the lexemes occurring in the vocabulary lists of the daughter languages. For example, assuming that the consonant cluster *šš* in Ojibwa is a reflex of either **hš* or **qš* in Proto-Algonquian, the proto-projections of Ojibwa *miššì* 'piece of firewood' would include **mihši* and **miqši*. The cognate identification process succeeds if the intersection of the sets of proto-projections generated from distinct daughter languages is not empty. The set intersection operation was implemented by alphabetically sorting all proto-projections. The potential cognate sets were subsequently analyzed by a linguist in order to determine whether they were in fact reflexes of the same proto-form and, if that was the case, to reconstruct the proto-form.

Hewson's method has a number of disadvantages. It is based exclusively on recurrent sound correspondences, with no recourse to potentially valuable phonetic and semantic information. It requires the user to provide a complete table of correspondences between daughter languages and the reconstructed proto-language. Since such a table of correspondences is established on the basis of multiple sets of confirmed cognates, the method is applicable only to language families that have already been throughly analyzed. In addition, the number of proto-projections increases combinatorially with the number of ambiguous reflexes that occur in a word. Anything less than a perfect match of correspondences may result in a cognate pair being overlooked.

Table 8. Examples of cognate pairs not included in Hewson's dictionary.

#	Lang.	Lexeme	Gloss	WordNet relation
1	Cree	*mōsāpēw*	'unmarried man'	*gloss synonymy*
	Men.	*mōsāpēwew*	'bachelor, single man'	
2	Fox	*kešēmanetōwa*	'great spirit'	*none*
	Men.	*kesēmanetōw*	'god'	
3	Cree	*wīhkēs*	'sweet-flag'	*keyword hypernymy*
	Ojib.	*wīkkēn'*	'iris'	
4	Men.	*enōhekan*	'pointer'	*none*
	Ojib.	*inō'ikan*	'that which is pointed at'	
5	Fox	*mīkātiweni*	'fight'	*gloss synonymy*
	Men.	*mīkātwan*	'war, fighting'	
6	Fox	*atāmina*	'maize-plant'	*keyword synonymy*
	Ojib.	*mantāmin*	'grain of corn'	
7	Fox	*ātesōhkākana*	'sacred story'	*keyword identity*
	Ojib.	*ātissōkkān*	'story or legend'	

Table 8 contains some interesting examples of Algonquian cognate pairs that are not found in Hewson's dictionary, but are recognized by the implementation of the method proposed in this paper. Their semantic, phonetic, and correspondence-based similarity considered in isolation may not be sufficient for their identification, but combining all three types of evidence results in a high overall similarity score. In particular, such pairs are bound to be missed by any approach that requires the identity of glosses as the necessary condition for consideration.

8 Conclusion

I have proposed a method of combining various types of evidence for the task of automatic cognate identification. In many cases, the new method achieves higher accuracy than the method based on the linear combination of scores. Moreover, the new method does not require manual parameter tuning, but instead can be trained on data from other language pairs.

The method proposed here is applicable both to structured (word lists) and unstructured (vocabulary lists) data. Apart from assisting the comparative linguists in proto-language reconstruction, it can be used to dramatically speed up the process of producing etymological dictionaries, even when little is known about the languages in question. The results of the experiments show that it is possible to discover a large number of cognates with good precision. To take the Fox–Menomini pair as an example, 70% recall at 50% precision signifies that the top 170 candidates contain 85 out of 121 existing cognate pairs. Moreover, many of the apparent false positives are in fact cognates or lexemes that are related in some way.

This paper belongs to a line of research that has already resulted in applications in such diverse areas as statistical machine translation [17] and the identification of confusable drug names [16]. In the long run, such applications may prove more important than the original linguistic motivation of the research that led to them. However, the language reconstruction framework is particularly well-suited for formulating the driving problems and for testing the proposed solutions.

Acknowledgments. Thanks to Colin Cherry, Russ Greiner, Dekang Lin, Dan Lizotte, and Dale Schuurmans for comments and advice. This research was supported by the Natural Sciences and Engineering Research Council of Canada.

References

1. Y. Al-Onaizan, J. Curin, M. Jahr, K. Knight, J. Lafferty, D. Melamed, F. Och, D. Purdy, N. Smith, and D. Yarowsky. Statistical machine translation. Technical report, Johns Hopkins University, 1999.

2. Chris Brew and David McKelvie. Word-pair extraction for lexicography. In K. Oflazer and H. Somers, editors, *Proceedings of the 2nd International Conference on New Methods in Language Processing*, pages 45–55, Ankara, Bilkent University, 1996.
3. Kenneth W. Church. Char_align: A program for aligning parallel texts at the character level. In *Proceedings of ACL-93: 31st Annual Meeting of the Association for Computational Linguistics*, pages 1–8, Columbus, Ohio, 1993.
4. Isidore Dyen, Joseph B. Kruskal, and Paul Black. An Indoeuropean classification: A lexicostatistical experiment. *Transactions of the American Philosophical Society*, 82(5), 1992.
5. Christiane Fellbaum, editor. *WordNet: an electronic lexical database*. The MIT Press, Cambridge, MA, 1998.
6. Jacques B. M. Guy. An algorithm for identifying cognates in bilingual wordlists and its applicability to machine translation. *Journal of Quantitative Linguistics*, 1(1):35–42, 1994. MS-DOS executable available at http://garbo.uwasa.fi.
7. John Hewson. Comparative reconstruction on the computer. In *Proceedings of the 1st International Conference on Historical Linguistics*, pages 191–197, 1974.
8. John Hewson. *A computer-generated dictionary of proto-Algonquian*. Hull, Quebec: Canadian Museum of Civilization, 1993.
9. John Hewson. Vocabularies of Fox, Cree, Menomini, and Ojibwa, 1999. Computer file.
10. Brett Kessler. *The Significance of Word Lists*. Stanford: CSLI Publications, 2001. Word lists available at http://spell.psychology.wayne.edu/∼bkessler.
11. Philipp Koehn and Kevin Knight. Knowledge sources for word-level translation models. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 27–35, 2001.
12. Grzegorz Kondrak. A new algorithm for the alignment of phonetic sequences. In *Proceedings of NAACL 2000: 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 288–295, 2000.
13. Grzegorz Kondrak. Identifying cognates by phonetic and semantic similarity. In *Proceedings of NAACL 2001: 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 103–110, 2001.
14. Grzegorz Kondrak. Determining recurrent sound correspondences by inducing translation models. In *Proceedings of COLING 2002: 19th International Conference on Computational Linguistics*, pages 488–494, 2002.
15. Grzegorz Kondrak. Identifying complex sound correspondences in bilingual wordlists. In *Proceedings of CICLing 2003: 4th International Conference on Computational Linguistics and Intelligent Text Processing*, pages 432–443, 2003.
16. Grzegorz Kondrak and Bonnie Dorr. Identification of confusable drug names: A new approach and evaluation methodology. 2004. In preparation.
17. Grzegorz Kondrak, Daniel Marcu, and Kevin Knight. Cognates can improve statistical translation models. In *Proceedings of HLT-NAACL 2003: Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 46–48, 2003. Companion volume.
18. Gideon S. Mann and David Yarowsky. Multipath translation lexicon induction via bridge languages. In *Proceedings of NAACL 2001: 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 151–158, 2001.
19. Tony McEnery and Michael Oakes. Sentence and word alignment in the CRATER Project. In J. Thomas and M. Short, editors, *Using Corpora for Language Research*, pages 211–231. Longman, 1996.

20. I. Dan Melamed. Automatic discovery of non-compositional compounds in parallel data. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 97–108, 1997.
21. I. Dan Melamed. Bitext maps and alignment via pattern recognition. *Computational Linguistics*, 25(1):107–130, 1999.
22. I. Dan Melamed. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249, 2000.
23. Michael P. Oakes. Computer estimation of vocabulary in protolanguage from word lists in four daughter languages. *Journal of Quantitative Linguistics*, 7(3):233–243, 2000.
24. Michel Simard, George F. Foster, and Pierre Isabelle. Using cognates to align sentences in bilingual corpora. In *Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 67–81, Montreal, Canada, 1992.
25. Morris Swadesh. Lexico-statistical dating of prehistoric ethnic contacts. *Proceedings of the American Philosophical Society*, 96:452–463, 1952.
26. Jörg Tiedemann. Automatic construction of weighted string similarity measures. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, College Park, Maryland, 1999.
27. Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173, 1974.
28. David Yarowsky and Richard Wincentowski. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of ACL-2000*, pages 207–216, 2000.

Term-Based Clustering and Summarization of Web Page Collections

Yongzheng Zhang, Nur Zincir-Heywood, and Evangelos Milios

Faculty of Computer Science, Dalhousie University
6050 University Ave., Halifax, NS, Canada B3H 1W5
{yongzhen,zincir,eem}@cs.dal.ca

Abstract. Effectively summarizing Web page collections becomes more and more critical as the amount of information continues to grow on the World Wide Web. A concise and meaningful summary of a Web page collection, which is generated automatically, can help Web users understand the essential topics and main contents covered in the collection quickly without spending much browsing time. However, automatically generating coherent summaries as good as human-authored summaries is a challenging task since Web page collections often contain diverse topics and contents. This research aims towards clustering of Web page collections using automatically extracted topical terms, and automatic summarization of the resulting clusters. We experiment with word- and term-based representations of Web documents and demonstrate that term-based clustering significantly outperforms word-based clustering with much lower dimensionality. The summaries of computed clusters are informative and meaningful, which indicates that clustering and summarization of large Web page collections is promising for alleviating the information overload problem.

1 Introduction

Effectively summarizing Web page collections becomes more and more critical as the online information continues to overload the World Wide Web. A concise and meaningful summary, which is generated automatically and consists of a few key phrases and key sentences, can help Web users understand the essential topics and main contents covered in the collection quickly without spending much browsing time [22].

However, automatically generating coherent summaries as good as human-authored summaries is a challenging task since Web page collections (e.g., Web sites of universities, organizations) often contain various topics and heterogenous contents. For example, a university's Web site may cover many topics such as courses, students, faculties and research. Summarizing the whole Web site often yields an incoherent summary or a summary that is heavily biased towards a subset of the topics included in the Web site. It is much more beneficial to first identify the essential topics in the Web site, and then summarize each individual topic.

Text clustering, which has been extensively studied in many scientific disciplines, plays an important role in organizing large amounts of heterogenous data into a small number of semantically meaningful clusters [15]. In particular, Web collection clustering is useful for summarization, organization and navigation of semi-structured Web pages [21].

Clustering of documents, including Web pages, suffers from the very high dimensionality of the feature vector space required if the naive bag-of-words representation of documents is used. In a high dimensional space, the distance between any two documents tends to be constant, making clustering on the basis of distance ill-defined [11]. Therefore the issue of reducing the dimensionality of the space is critical. Automatic term extraction [9], based on a combination of linguistic and statistical constraints, has the promise of leading to reduced dimensionality representations by focusing on semantically meaningful word collocations [13].

This research aims towards K-Means and EM clustering of Web page collections using automatically extracted topical terms, and automatic summarization of the resulting clusters. Summarization of a cluster proceeds with the identification and extraction of keyterms and key sentences in the *narrative* text, which constitute the summary of the cluster [22].

The quality of clustering with term-based document representation is compared to that with word-based document representation, which is used as the baseline method. We demonstrate that term-based clustering significantly outperforms word-based clustering with much lower dimensionality. The significance of Web collection clustering for automatic Web collection summarization is also investigated.

The rest of the paper is organized as follows. Section 2 reviews published Web collection clustering and summarization approaches, and Section 3 explains how to construct a Web page collection for experimental purposes. Section 4 describes how to conduct K-Means and EM clustering with both word- and term-based document representations and Section 5 evaluates the clustering quality. Section 6 discusses automatic summarization of resulting clusters. Finally, Section 7 concludes our work and describes future research directions.

2 Related Work

A variety of approaches to text clustering have been developed [2,12,15]. Typically clustering approaches can be categorized as *agglomerative* or *partitional* based on the underlying methodology of the algorithm, or as *hierarchical* or *flat* (non-hierarchical) based on the structure of the final solution [23].

In general, text clustering involves constructing a vector space model and representing documents by feature vectors. First, a set of features (e.g., bag of words) is properly selected from the document corpus. Second, each document is represented by a feature vector, which consists of weighting statistics of all features. Finally, clustering proceeds by measuring the similarity (e.g., a function of

Euclidean distance) between documents and assigning documents to appropriate clusters.

One important issue for effective clustering is feature selection. A good feature set is effective if it can discriminate dissimilar documents as much as possible and its dimensionality is as low as possible. Approaches to Web collection clustering have been either *word-based* or *link-based.* Word-based approaches [4,5] use a set of common words shared among documents as features, which suffers from the high space dimensionality while link-based approaches [6,20] analyze the hyperlinks between Web documents for feature selection, which depends on the availability of co-citations. Some Web page clustering systems [14,21] use a combination of above two.

Research work in [13] demonstrates that automatically extracted topical terms can reduce the space dimensionality significantly while providing comparable performance with link-based feature representation (citation graph) in an information retrieval task, i.e., document similarity analysis. We aim to use automatically extracted terms for Web collection clustering and compare the clustering quality to that with bag of words representation.

Zhang et al. [22] propose an effective content-based approach to automatic Web site summarization, which can be applied to automatic summarization of computed clusters. Traditional text summarization systems are mostly *extraction-based* via the identification of the most significant sentences in source text for summary generation [10]. However, Web pages often contain diverse textual fragments such as bullets (which are not complete phrases and therefore difficult to analyze linguistically) or short phrases, that carry no information (e.g., date page last revised / copyright note), so being able to identify narrative text apart from non-narrative text is very important when moving from traditional coherent text such as news stories to Web documents. By filtering out non-narrative text identified by decision tree rules, Web-based applications, including automatic term extraction and Web collection clustering, can make use of state-of-the-art techniques for cleaner text.

3 Web Page Collection

The Text REtrieval Conference (TREC[1]) provides a Web test collection .GOV[2] with 18G data crawled from the .GOV Web site[3], which is a huge site with 14 main topics[4]. Each topic has a *description* Web page which lists all the subtopics categorized in the current topic. In order to construct a Web page collection for testing the clustering quality using word- and term-based document representations, a Web crawler is designed to collect Web pages of subtopics from each of the 14 topics. The 14 Web pages of topic description are excluded.

[1] http://trec.nist.gov
[2] http://www.ted.cmis.csiro.au/TRECWeb/govinfo.html
[3] http://www.firstgov.gov
[4] http://www.firstgov.gov/Citizen/Topics/All_Topics.shtml

A total of 1123 Web pages of subtopics is collected. Among them, a subtotal of 141 link-broken and application files (.pdf, .ps, .doc and other binary) is removed, leading to a smaller set of 982 Web pages. Then plain text from all Web pages is extracted by the text browser Lynx[5], which is found to outperform several alternative text extraction tools such as *HTML2TXT*[6] and *html2txt*[7] [22]. Text files with more than $100K$ size are truncated to that size.

The final data set used for clustering experiments is summarized in Table 1, where i is the topic index and $|T_i|$ is the number of Web pages in each topic T_i.

Table 1. Number of Web pages in each of 14 topics.

i	**Topic** T_i	$\lvert T_i \rvert$
1	Benefits and Grants	58
2	Consumer Protection	56
3	Defense and International	86
4	Education and Jobs	89
5	Environment, Energy and Agriculture	101
6	Family, Home and Community	119
7	Health	94
8	History, Arts and Culture	75
9	Money and Taxes	55
10	Public Safety and Law	58
11	Public Service and Volunteerism	69
12	Science and Technology	36
13	Travel and Recreation	48
14	Voting and Elections	38

As shown in Table 1, the most populated topic is *Family, Home and Community* with 119 Web pages while the least populated topic is *Science and Technology* with only 36 Web pages.

4 *K*-means and EM Clustering

In this section we discuss how to conduct word- and term-based K-Means and EM clustering on the Web page collection constructed in Section 3.

4.1 *K*-means and EM Algorithms

The K-Means algorithm has been widely used due to its implementation simplicity. This non-hierarchical method first selects K data points as the initial

[5] http://lynx.isc.org
[6] http://user.tninet.se/~jyc891w/software/html2txt/
[7] http://cgi.w3.org/cgi-bin/html2txt

centroids, one for each of K clusters. Second, all data points are assigned to the cluster whose centroid is the closest to the current data point. Third, the centroid of each cluster is recalculated. Steps two and three are repeated until the centroids do not change [18].

The Expectation Maximization (EM) algorithm was first introduced by Dempster et al. [7] as an approach to estimating the missing model parameters in the context of fitting a probabilistic model to data. It can be seen as an iterative approach to optimization. It first finds the expected value of the *log likelihood* with respect to the current parameter estimates. The second step is to maximize the expectation computed in the first step. These two steps are iterated if necessary. Each iteration is guaranteed to increase the log likelihood and the algorithm is guaranteed to converge to a local maximum of the likelihood function [1]. EM has many applications including text classification and text clustering. It is one of the most popular statistical unsupervised learning algorithms.

4.2 Word- and Term-Based Feature Selection

As discussed in Section 2, feature selection plays an important role in achieving high quality clustering. In our work, we experiment with two ways of feature selection, i.e., bag of words (word-based) and automatically extracted terms (term-based). In both cases, the text parts of Web pages are represented using the standard normalized TF-IDF weighting scheme, which is used in many text-based applications [8,16].

Word-based Representation of Documents. The naive bag of words method selects a set of common words shared among documents as features. First, a standard list of 425 stopwords [8] is removed from each text file. Second, each unique word is stemmed using the Porter stemmer, and its frequency in the current text file (TF part) and the number of documents where it appears (IDF part) are recorded. Finally, the word list is sorted in descending order of total frequency in all documents. It is observed there is a total of 39245 unique words in 982 text files. The most frequent word *state* appears 5816 times while about 6000 words appear only once. After we remove words which appear less than 6 times in the corpus, there are 5526 words left in the list.

In selecting the words to be used as features, it is observed that words appearing in most documents in the corpus are not useful features, because they do not help discriminate among clusters. So we re-rank the list of 5526 words according to their IDF in order to set proper upper and lower cutoffs for selecting words appearing with intermediate frequency [13]. After the words used as features are selected, their TF-IDF values in each document are calculated and normalized as in (1) and (2), respectively.

$$w_{i,j} = f_{i,j} \times \log \frac{N}{df_i}. \tag{1}$$

$$W_{i,j} = \frac{w_{i,j}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2}}. \tag{2}$$

where, $w_{i,j}$ is the TF-IDF weight of word i in document j; $f_{i,j}$ is the frequency of word i in document j; N is the total number of documents in the collection, i.e., 982; df_i is the number of documents containing word i; and $W_{i,j}$ is the normalized weight of word i in document j.

We experimented with different word cutoffs such as [10, 4000] (from the 10^{th} word to the 4000^{th} word ranked by IDF) and found that the best word cutoff with respect to clustering quality (evaluated by F-measure in Section 5) is [15, 3000].

Term-based Representation of Documents. Word-based representation involves thousands of features and subsequent clustering often suffers from the dimensionality curse [11]. Multi-word terms are known to be linguistic descriptors of documents. Automatic term extraction is a useful tool for many text related applications such as text clustering and document similarity analysis. It is beneficial to automatically extract multi-word terms as features in order to significantly reduce the high dimensionality [13].

In our work, we apply a state-of-the-art method, *C-value/NC-value* [9], to extract multi-word terms from the Web page collection automatically. The *C-value* is a domain-independent method used to automatically extract multi-word terms. It aims to get more accurate terms than those obtained by the pure frequency of occurrence method, especially terms that may appear as nested within longer terms. The *NC-value* is an extension to *C-value*, which incorporates context words information into term extraction. Context words are those that appear in the vicinity of candidate terms, i.e. nouns, verbs and adjectives that either precede or follow the candidate term. This term extraction approach consists of both linguistic analysis (linguistic filter, part-of-speech tagging [3], and stop list [8]) and statistical analysis (frequency analysis, *C-value/NC-value*). Experiments in [9,13] show that *C-value/NC-value* method performs well on a variety of special text corpora.

However, unlike traditional text documents with well-structured discourse, Web documents often contain diverse textual segments (e.g., bullets and short phrases) which are not suitable for extraction of semantically meaningful terms. We apply the same approach as proposed in [22] to identify *narrative* paragraphs in the Web collection and consequently terms are extracted from narrative text only. First, text parts of Web pages are automatically segmented into paragraphs by Lynx. Second, one classifier, LONG, is used to filter out *short* paragraphs. Third, the other classifier, NARRATIVE, is in turn used to extract *narrative* paragraphs from *long* paragraphs identified in the previous step. These two classifiers are trained by the decision tree tool C5.0[8] based on features extracted by shallow natural language processing.

[8] http://www.rulequest.com/see5-unix.html

Then the *C-value/NC-value* method is applied and 1747 terms are extracted. These terms are ranked by NC-value and an NC-value threshold 17.5, which is 5% of the maximum NC-value, is set. This in return produces a term list, C, of 924 terms.

It is observed that the term list C contains some noun phrases (e.g., *home page*), which appear frequently in Web pages. These noun phrases can be treated as *Web-specific* stop words [17] and must be removed. We experimented with 60 DMOZ[9] Web sites and manually identified a stop list, L, of 81 noun phrases (e.g., *Web site*, *home page*, *credit card*, *privacy statement*, ...). The term list C is filtered through the noun phrase stop list L and the resulting term list T contains 892 terms.

Next, all documents are scanned in order to record the TF and IDF values for all terms in T. As we did in the word-based approach, we re-rank all the terms in T according to their IDF in order to set proper upper and lower cutoffs for selecting terms. We experiment with different term cutoffs and the best term cutoff is found to be [1, 450].

5 Clustering Experiments and Evaluation

In this section we discuss the clustering experiments and evaluate the quality of the clustering results.

In our work, Java programs of K-Means and EM algorithms in the WEKA[10] package are used. For each cutoff (see previous section) of words and terms, we construct either a document-word matrix or a document-term matrix with Attribute-Relation File Format (ARFF)[11], where each document is represented with normalized TF-IDF values of either words or terms. Then we run the clustering software (we set $K = 14$ and force EM to create 14 clusters) and compare the clustering quality using word- and term-based document representations.

5.1 Evaluation Schemes

There are two main types of measures of clustering quality, i.e., *internal quality measure* and *external quality measure*. Internal quality measure uses the "overall similarity", which is based on the pairwise similarity of documents in a cluster, to evaluate the clustering quality. It compares different sets of clusters without reference to external knowledge. On the other hand, external quality measure such as *entropy* and *F-measure* examines the clustering quality by comparing the resulting clusters to known classes (topics) [18].

We use the F-measure to evaluate the clustering quality since topic information for all Web pages is available. F-measure combines *precision* and *recall*. For a particular cluster C_i, we have to decide which topic T_j it most probably

[9] http://dmoz.org
[10] http://www.cs.waikato.ac.nz/~ml/weka/index.html
[11] http://www.cs.waikato.ac.nz/~ml/weka/arff.html

belongs to. More specifically, we calculate the precision $P_{i,j}$, recall $R_{i,j}$, and F-measure $F_{i,j}$ of cluster C_i with respect to topic T_j as in (3).

$$\begin{aligned} P_{i,j} &= \frac{n_{i,j}}{|C_i|},\ \ R_{i,j} = \frac{n_{i,j}}{|T_j|}, \\ F_{i,j} &= \frac{2 \cdot P_{i,j} \cdot R_{i,j}}{P_{i,j} + R_{i,j}}\ \ (i, j \in [1..14]). \end{aligned} \tag{3}$$

where, $n_{i,j}$ is the number of documents in topic T_j which appear in cluster C_i; $|C_i|$ is the number of documents in cluster C_i; and $|T_j|$ is the number of documents in topic T_j. Then the cluster C_i will belong to the topic T_j which maximizes $F_{i,j}$. This is formally represented in (4).

$$F_i = \max_{j=1}^{14} F_{i,j}\ \ (i \in [1..14]). \tag{4}$$

Finally the overall value of F-measure F is calculated by taking a weighted average of all F_i values as represented in (5).

$$F = \sum_{i=1}^{14} \frac{|T_i|}{N} F_i,\ \ N = \sum_{j=1}^{14} |T_j|. \tag{5}$$

5.2 Word-Based Clustering Quality

We use both K-Means and EM algorithms to experiment with various document-word matrices, which are constructed on different word cutoffs. Table 2 summarizes the top 5 overall F values obtained with both algorithms. It is observed that EM algorithm achieves better F values than K-Means algorithm in all 5 word cutoffs.

Table 2. The top 5 F values achieved by word-based K-Means and EM clustering.

Cutoff_{word}	[15, 3000]	[10, 3000]	[10, 3500]	[15, 4000]	[5, 3000]
$F_{K-Means}$	0.52	0.51	0.46	0.44	0.42
F_{EM}	0.59	0.57	0.50	0.49	0.43

As shown in Table 2, both K-Means and EM achieve better F values with the word cutoff [15, 3000] than with the word cutoff [10, 3000]. This indicates that the words at the top of the word list (ranked by IDF) appear too frequently and they are not appropriate for feature selection. Also cutoff [10, 3000] outperforms cutoff [10, 3500], which indicates that the words at the bottom of the word list appear too rarely and they are not good features either. This demonstrates our assumption that words with intermediate document frequency are better choices of features.

5.3 Term-Based Clustering Quality

Document-term matrices are constructed on different term cutoffs. Then both K-Means and EM algorithms are applied to these matrices. Table 3 summarizes the top 5 overall F values achieved with various term cutoffs. We observe that EM achieves bigger F values than K-Means in all 5 term cutoffs.

Table 3. The top 5 F values achieved by term-based K-Means and EM clustering.

Cutoff$_{term}$	[1, 450]	[1, 500]	[2, 450]	[2, 500]	[3, 500]
$F_{K-Means}$	0.67	0.67	0.67	0.65	0.64
F_{EM}	0.68	0.68	0.68	0.67	0.65

As shown in Table 3, there is no difference between the F values using either K-Means or EM with the first 3 term cutoffs. This indicates that a proper number (e.g. 450) of terms at the top of the term list (ranked by IDF) can be readily used as features.

Furthermore, we do EM clustering by fixing the lower term cutoff to 1, and changing the upper term cutoff with different values varying from 300 to 892 (full size of the term list). As shown in Fig. 1, different upper term cutoffs lead to different F values. The best F value is achieved with the upper term cutoff 450. If we increase or decrease the upper term cutoff such as using 350 and 600, then the clustering performance becomes worse. This indicates that there exists an optimal dimensionality for term-based clustering. Moreover, it is observed that with cutoffs [1, 300] and [1, 315], EM clustering fails because *null* feature vector appears, i.e., there is some document which does not include any one of the terms. This indicates that there is a minimum value for the upper term cutoff for valid clustering.

Likewise, we do EM clustering by fixing the upper term cutoff to 450, and changing the lower term cutoff with values varying from 1 to 10. It is observed that there is no much difference between resulting F values and that a lower term cutoff of 6 will lead to invalid clustering, i.e., there is at least one null vector in this case. This indicates that with much lower dimensionality in term-based clustering, terms at the top of the term list are indispensable.

The experiments above are a rough approximation of the search for optimal upper and lower term cutoffs. We leave this two-dimension search problem for future research.

5.4 Comparison of Word- and Term-Based Clustering

The main objective of our work is to investigate if term-based clustering can significantly outperform word-based clustering on Web page collections.

Table 4 summarizes the details of the best F values in word- and term-based K-Means clustering, where w_i is the weight of each topic, i.e., $\frac{|T_i|}{N}$, and F is the

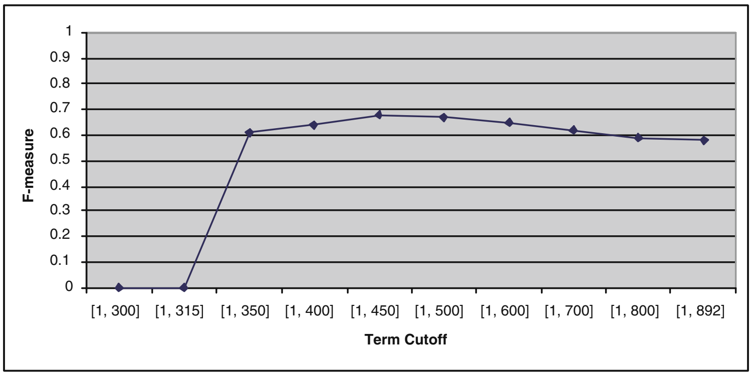


Fig. 1. Varying F values corresponding to different upper term cutoffs with fixed lower term cutoff 1 in EM clustering.

sum of $w_i \cdot F_i$ over all 14 topics, as calculated in (5). K-Means algorithm achieves the best overall F values, 0.52 using the word cutoff [15, 3000], and 0.67 using the term cutoff [1, 450], respectively.

Table 4. The best F values achieved in K-Means clustering using the word cutoff [15, 3000] and the term cutoff [1, 450], respectively.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	F
w_i	0.06	0.06	0.09	0.09	0.10	0.12	0.10	0.08	0.06	0.06	0.07	0.04	0.05	0.04	—
$F_{i_{word}}$	0.53	0.45	0.67	0.43	0.39	0.68	0.33	0.54	0.44	0.64	0.61	0.49	0.70	0.45	**0.52**
$F_{i_{term}}$	0.62	0.59	0.83	0.55	0.71	0.82	0.53	0.59	0.61	0.71	0.66	0.49	0.87	0.63	**0.67**

In order to measure if the difference between the quality of word- and term-based clustering (i.e., $w_i \cdot (F_{i_{term}} - F_{i_{word}})$ $(i = 1..14)$) is significant, we apply the standard two-tail paired t-test, which generally compares two different methods used for experiments carried in pairs.

The t-test carried on Table 4 obtains a t-statistic equal to 4.543. By checking the t-table, we have $t_{0.05,13} = 2.160$. Since $t > t_{0.05,13}$ (P-value < 0.001), there is a significant difference between the clustering quality of word- and term-based document representation using K-Means algorithm. More precisely, term-based K-Means clustering significantly outperforms word-based K-Means clustering with much lower dimensionality.

Table 5 shows the details of the best F values in word- and term-based EM clustering. EM algorithm achieves the best overall F values, 0.59 using the word cutoff [15, 3000], and 0.68 using the term cutoff [1, 450], respectively. Similar t-test analysis as above obtains a t-statistic equal to 3.237 and a P-value less than 0.01, which shows that term-based EM clustering significantly outperforms word-based EM clustering.

Table 5. The best F values achieved in EM clustering using the word cutoff [15, 3000] and the term cutoff [1, 450], respectively.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	F
$F_{i_{word}}$	0.63	0.54	0.65	0.47	0.61	0.76	0.48	0.68	0.43	0.52	0.55	0.38	0.77	0.69	**0.59**
$F_{i_{term}}$	0.75	0.89	0.54	0.49	0.66	0.84	0.51	0.77	0.64	0.59	0.58	0.54	0.91	0.78	**0.68**

Moreover, with the term cutoff [1, 335], EM algorithm achieves an overall F value 0.59, which is equal to the best F value with the word cutoff [15, 3000]. This indicates that term-based document representation can significantly reduce the dimensionality by almost an order of magnitude while maintaining comparable performance with word-based document representation in EM clustering.

5.5 Comparison of *K*-means and EM Clustering

We also apply t-test to measure if there is a significant difference between clustering quality of K-Means and EM using the best word and term cutoffs.

The t-test for comparison of EM and K-Means algorithms using the best word cutoff [15, 3000] obtains a t-statistic equal to 2.199 and a P-value less than 0.05, so there is a significant difference between the quality of word-based EM clustering and word-based K-Means clustering. More precisely, the EM algorithm significantly outperforms the K-Means algorithm using word-based document representation. However, a similar t-test shows that there is no significant difference (t-statistic = 0.042, P-value = 0.967) between clustering quality of K-Means and EM when using the best term cutoff [1, 450].

Regarding the computational complexity, it is observed that K-Means clustering is roughly seventeen times (CPU time: 37 seconds vs. 630 seconds) faster than EM clustering with the best term cutoff when running in a Linux machine.

6 Automatic Cluster Summarization

In this section we explain how to summarize each individual cluster obtained by EM clustering using the best term cutoff [1, 450] described above and evaluate cluster summaries.

6.1 Cluster Summary Generation

The automatic cluster summarization is a multi-step process similar with the keyword-based approach proposed by Zhang et al. [22].

1. First a list of multi-word terms is extracted by the *C-value*/*NC-value* method from the narrative text of Web pages in each cluster. Then the term list is filtered through the noun phrase stop list L and the top 25 terms (ranked by *NC-value*) are kept as keyterms.
2. Narrative text of the Web pages in the current cluster is scanned to extract the most significant sentences. The significance of a sentence is measured by calculating a weight value, which is the maximum of weights of *word clusters* within the sentence. A word cluster is defined as a list of consecutive words which starts and ends with a keyterm and at most 2 non-keyterms must separate any two neighboring keyterms. The weight of a word cluster is calculated as the sum of keyterm weights divided by the number of keyterms in the word cluster.
3. A summary is generated, which consists of the top 25 keyterms and the top 5 key sentences. The numbers 25 and 5 are determined by both the informativeness of the keyterms and the key sentences and the size of the cluster summaries.

As an example, Table 6 gives the cluster summary corresponding to Topic 7: Health. As we can see, this summary provides informative contents of the Health topic.

Table 6. An example of cluster summary corresponding to the *Health* topic.

Part I. top 25 keyterms
health administration, veterans health, health care, drug administration, food safety, veterans health administration, severe acute respiratory syndrome, health maintenance organization, mine safety, veterans health administration facilities, health insurance, vital events, drug abuse, veterans integrated service networks, veterans health administration program, health information, acute respiratory syndrome, respiratory syndrome, severe acute, women health, other communication disorder, health care professionals, health administration facilities, health maintenance, health service
Part II. top 5 key sentences
1. The Veterans Health Administration (VHA) provides a broad spectrum of medical, surgical, and rehabilitative care to its customers. 2. VHA Organizations A number of offices work together to make the Veterans Health Administration (VHA) an efficient and patient-centered health care system. 3. The Mine Safety and Health Administration - Look here for information about the dangers of playing near or in mines. 4. Data on health status, healthy lifestyles, illness and disability, the use of health care and vital events. 5. The Food and Drug Administration (FDA) today is advising women and health care professionals about important new safety changes to labeling of all estrogen and estrogen with progestin products for use by postmenopausal women.

6.2 Cluster Summary Evaluation

In this subsection, we describe how to evaluate the quality of cluster summaries in a way which has been extensively used in related work [13,19]. Human subjects are asked to read each cluster summary and judge the relatedness of keyterms and key sentences to the corresponding topic as follows:

1. For each cluster, browse the description Web page (see Section 3) of its corresponding topic (which maximizes the F value of the current cluster) for a sufficient time in order to understand the subtopics categorized in the current topic.
2. Read the cluster summary and rank each *summary item*, i.e., keyterm or key sentence, into *good*, *fair* or *bad* using the following rules:
   - If it is strongly pertinent to the current topic, rank it *good*.
   - If it is pertinent to the topic, rank it *fair*.
   - If it is not pertinent to the topic, rank it *bad*.
3. Count n_g, n_f, and n_b, which is the number of good, fair, and bad items in each summary, respectively.

Related research in [19] defines *acceptable* terms as good and fair terms. Let p_t and p_s be the percentage of acceptable keyterms and key sentences, respectively. We formally define p_t and p_s in (6).

$$p_t, p_s = \frac{n_g + n_f}{n_g + n_f + n_b}. \tag{6}$$

For example, in the summary example shown in Table 6, there are 17 good, 7 fair, and 1 bad keyterms; 2 good, 3 fair, and 0 bad key sentences. So the percentage of acceptable keyterms is calculated as $\frac{17+7}{25} = 96\%$, and the percentage of acceptable key sentences is 100%.

Table 7 summarizes the percentage of acceptable keyterms and key sentences for all cluster summaries. Our approach achieves an average of 83.7% acceptable keyterms and 88.6% acceptable key sentences, which indicates that the cluster summaries are acceptable to human readers.

Table 7. Details of summary quality values.

Summary	1	2	3	4	5	6	7	8	9	10	11	12	13	14	**Average**
p_t(%)	100	96	64	96	56	92	96	68	96	84	84	72	80	88	83.7
p_s(%)	100	100	80	100	100	60	100	100	100	80	80	60	80	100	88.6

7 Conclusion and Discussion

In this paper, we experiment with word- and term-based representations of Web documents for K-Means and EM clustering. The term-based approach relies on a start-of-the-art automatic term extraction method on narrative content of the Web page collection. We evaluate the difference between word- and term-based clustering and demonstrate that term-based clustering significantly outperforms word-based clustering with much lower dimensionality. We also show that summarization of individual clusters provides a good overview of the various essential topics covered in the Web page collection and this approach is critical for alleviating the information overload problem on the World Wide Web.

Future research involves several directions: 1) Investigation of automatic Web-specific stop words generation [17] to achieve better term extraction from Web page collections; 2) Search for optimal upper and lower term cutoffs for best clustering quality; 3) Bisecting K-Means and EM clustering to produce a hierarchical decomposition of the essential topics and their subtopics covered in the data collection; 4) Application on the whole .GOV collection to estimate the effectiveness and efficiency of our approach on the huge data set.

Acknowledgements. We are thankful to reviewers for many valuable suggestions on this work. The research has been supported by grants from the Natural Sciences and Engineering Research Council of Canada.

References

[1] J. Bilmes. A Gentle Tutorial on the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. Technical Report ICSI-TR-97-021, University of California, Berkeley, 1997.

[2] D. Boley. Principal Direction Divisive Partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

[3] E. Brill. A Simple Rule-based Part of Speech Tagger. In *Proceedings of the 3rd Conference on Applied Natural Language Processing (ANLP-92)*, pages 152–155, 1992.

[4] A. Broder, S. Glassman, M. Manasse, and G. Zweig. Syntactic Clustering of the Web. In *Proceedings of the 6th International World Wide Web Conference*, pages 391–404, 1997.

[5] D. Cutting, D. Karger, J. Pedersen, and J. Tukey. Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318–329, 1992.

[6] J. Dean and M. Henzinger. Finding Related Pages in the World Wide Web. In *Proceedings of the 8th International World Wide Web Conference*, pages 389–401, 1990.

[7] A. Dempster, N. Laird, and D. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[8] W. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*. Prentice Hall, Englewood Cliffs, NJ, 1992.

[9] K. Frantzi, S. Ananiadou, and H. Mima. Automatic Recognition of Multiword Terms. *International Journal of Digital Libraries*, 3(2):117–132, 2000.

[10] J. Goldstein, V. Mittal, J. Carbonell, and J. Callan. Creating and Evaluating Multi-document Sentence Extract Summaries. In *Proceedings of the 9th ACM International Conference on Information and Knowledge Management (CIKM'00)*, pages 165–172, 2000.

[11] E. Han, G. Karypis, V. Kumar, and B. Mobasher. Clustering in a High-dimensional Space Using Hypergraph Models. Technical Report TR-97-063, Department of Computer Science and Engineering/Army HPC Research Center, University of Minnesota, 1997.

[12] A. Hotho, A. Maedche, and S. Staab. Ontology-based Text Clustering. In *Proceedings of the IJCAI Workshop on "Text Learning: Beyond Supervision"*, 2001.

[13] E. Milios, Y. Zhang, B. He, and L. Dong. Automatic Term Extraction and Document Similarity in Special Text Corpora. In *Proceedings of the 6th Conference of the Pacific Association for Computational Linguistics (PACLing'03)*, pages 275–284, 2003.

[14] D. Modha and W. Spangler. Clustering Hypertext with Applications to Web Searching. In *Proceedings of the 11th ACM Conference on Hypertext and Hypermedia*, pages 143–152, 2001.

[15] H. Schutze and H. Silverstein. Projections for Efficient Document Clustering. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97)*, pages 74–81, 1997.

[16] F. Sebastiani. Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 34(1):1–47, March 2002.

[17] M. Sinka and D. Corne. Towards Modernized and Web-Specific Stoplists for Web Document Analysis. In *Proceedings of IEEE/WIC International Conference on Web Intelligence (WI'03)*, pages 396–402, 2003.

[18] M. Steinbach, G. Karypis, and V. Kumar. A Comparison of Document Clustering Techniques. In *Proceedings of KDD Workshop on Text Mining*, 2000.

[19] P. Turney. Coherent Keyphrase Extraction via Web Mining. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, pages 434–439, 2003.

[20] Y. Wang and M. Kitsuregawa. Use Link-based Clustering to Improve Web Search Results. In *Proceedings of the 2nd International Conference on Web Information Systems Engineering (WISE'01)*, pages 115–124, 2001.

[21] Y. Wang and M. Kitsuregawa. Evaluating Contents-link Coupled Web Page Clustering for Web Search Results. In *Proceedings of the 11th ACM International Conference on Information and Knowledge Management (CIKM'02)*, pages 499–506, 2002.

[22] Yongzheng Zhang, Nur Zincir-Heywood, and Evangelos Milios. Summarizing Web Sites Automatically. In *Proceedings of the 16th Conference of the Canadian Society for Computational Studies of Intelligence (AI'03)*, pages 283–296, 2003.

[23] Y. Zhao and G. Karypis. Evaluation of Hierarchical Clustering Algorithms for Document Datasets. In *Proceedings of the 11th ACM International Conference on Information and Knowledge Management (CIKM'02)*, pages 515–524, 2002.

The Frequency of Hedging Cues in Citation Contexts in Scientific Writing

Robert E. Mercer[1], Chrysanne Di Marco[2], and Frederick W. Kroon[2]

[1] University of Western Ontario, London, Ontario, N6A 5B7,
mercer@csd.uwo.ca

[2] University of Waterloo, Waterloo, Ontario, N2L 3G1,
{cdimarco,fwkroon}@uwaterloo.ca

Abstract. Citations in scientific writing fulfill an important role in creating relationships among mutually relevant articles within a research field. These inter-article relationships reinforce the argumentation structure that is intrinsic to all scientific writing. Therefore, determining the nature of the exact relationship between a citing and cited paper requires an understanding of the rhetorical relations within the argumentative context in which a citation is placed. To determine these relations automatically in scientific writing, we have suggested that stylistic and rhetorical cues will be significant. One type of cue that we have studied is the discourse cue, which provides cohesion among textual components. Another form of rhetorical cue involves hedging to modify the affect of a scientific claim. Hedging in scientific writing has been extensively studied by Hyland, including cataloging the pragmatic functions of the various types of cues. In this paper we show that the hedging cues proposed by Hyland occur more frequently in citation contexts than in the text as a whole. With this information we conjecture that hedging cues are an important aspect of the rhetorical relations found in citation contexts and that the pragmatics of hedges may help in determining the purpose of citations.

1 Introduction

1.1 Why We Are Studying Hedging

Citations in scientific writing are used to connect mutually relevant articles within a research field. A citation index, which is a compilation of connections between citing and cited articles, provides a means to navigate the collection of scientific articles. Typically, a navigational foray into this collection is overwhelmed by the sheer numbers of related articles. However, users of citation indexes usually have a more focussed purpose than merely to find related articles. Very often, the index search could be narrowed if the writer's purpose for generating the citation were to label the citation connection.

To label citations with a citation function drawn from a list of possible functions requires an analysis of the text surrounding the citation coupled with the knowledge that scientific researchers want to communicate their results and want to argue that these results become part of the body of scientific knowledge. This latter aspect has been extensively studied by rhetoricians of science, e.g. [5], [6], [12].

These rhetorical stances are communicated with a variety of stylistic effects. Fundamental to our work is the claim that these stylistic effects may be realized through surface cues in text thereby enabling the reader to interpret the rhetorical nature of the text appropriately. With this in mind, our overall task is to map stylistic cues to rhetorical relations. Our previous work has described the importance of *discourse cues* in enhancing inter-article cohesion signalled by citation usage [11]. We have also begun to compile a catalogue of fine-grained discourse cues that exist in citation contexts [1]. In the following we investigate another class of pragmatic cues signalled by surface means—called 'hedging'—that can be exploited to uncover the function of citations in scientific writing.

The hypothesis that we test is that hedging cues play an important role in the rhetoric that is intrinsic to citation usage. We have investigated this hypothesis by doing a frequency analysis of hedging cues in citation contexts in a corpus of 985 biology articles. We have obtained statistically significant results indicating that hedging is used more frequently in citation contexts than the text as a whole. Given the presumption that writers make stylistic and rhetorical choices purposefully, we propose that we have further evidence that connections between fine-grained linguistic cues and rhetorical relations exist and that these may be catalogued. Further, the rhetorical relation can then suggest a citation function.

1.2 Why Hedging Is Used in Scientific Writing

The 'job' of a scientific researcher is usually thought to be focussed primarily on the discovery of new factual knowledge. However, a large and critical component of scientific discovery concerns the acceptance of new results by the research community and their ultimate integration into the community's archival knowledge. The scientific process is therefore not only about *doing* science but about *persuading* others of the validity of results so that these may be judged worthy of publication. Along with the validation and publication of results, the reputation of the individual researcher becomes enhanced, with concomitant effects on the person's standing in her community, chances of receiving tenure or promotion, and likely success in subsequent grant reviews.

A variety of rhetorical strategies may be used in scientific writing to create both a sense of social involvement in the scientific community and a persuasive influence in promoting the reader's acceptance of the claims being made. The rhetorical means through which an author achieves this persuasive purpose may take various forms: *hedging*, to weaken the assertiveness of new claims, so they may be more readily judged acceptable by reviewers; *citations*, to indicate a network of mutually supportive or contrasting works; and *politeness* to build social closeness with other researchers [8] (p. 64). Of these, the rhetorical strategies that will concern us here are the uses of hedging and citations, and the extent to which these two strategies are linked.

Hedging strategies may be described as "...the linguistic devices used to qualify a speaker's confidence in the truth of a proposition, the kind of caveats like *I think*, *perhaps*, *might*, and *maybe* which we routinely add to our statements to avoid commitment to categorical assertions. Hedges therefore express tentativeness and possibility in communication, and their appropriate use in scientific discourse is critical [8] (p. 1)". The use of hedging in scientific writing is actually part of a larger pragmatic purpose on the part of the author: she is simultaneously putting forth claims that must be seen as worthy of publication or as a basis for funding, while at the same time she must be careful to present her work as acceptable to her social community of academic peers and as constituting a continuation of established knowledge (even though she may be to some extent challenging or replacing previous work). Hyland [8] (p. 196) describes a text in which the writer has proposed a radical explanation for a process that is a core issue in her research area. As he analyzes the text, he points out how the writer goes even further, in making serious challenges to current theories. Not only is the writer concerned about supporting her own scientific claim, Hyland observes, but with protecting her position in her research community: "In making this proposal, the writer implicitly attributes serious inadequacies in current theories in their interpretations of critical data. She therefore runs the very real risk of having the claim rejected by a community of peers who, she perceives, have a great deal invested in the existing view and who are likely to defend it without giving serious consideration to her work" (p. 196). To address these conflicting pragmatic purposes, the paper is thick with hedges: modal verbs and adverbs, epistemic lexical verbs, indefinite quantifiers, and admissions of limiting conditions, all contriving to "[create] a rhetorical and interpersonal context which seeks to pre-empt the reader's rejection" [8] (p. 196).

In attempting to persuade readers of a research article that the results—*knowledge claims*—contained therein constitute a valuable addition to the discipline, the author is in effect engaging in an intricate 'dialogue' with her audience. Hedging can thus be viewed as a type of modality that allows

> a form of participation by the speaker in the speech event. Through modality, the speaker associates with the thesis an indication of its status and validity in his own judgement; he intrudes, and takes up a position. [7] (p. 335), quoted in [8] (p. 47)

We may say therefore that hedging as a rhetorical technique in building up a scientific argument is intrinsic to scientific writing. Further, the pragmatic functions of hedging, conveying persuasive effect to enhance new knowledge claims and aiding the writer in building a social context with the reader, would seem to indicate that effectively managing the use of hedging is essential to the scientific process. As Hyland [8] states in his review of hedging strategies in scientific writing:

> Hedging is critical in scientific discourse because it helps gain communal acceptance for knowledge. Scientific 'truth' is as much a social as an intellectual category, and the distinction writers make between their

subject matter and how they want readers to understand their relationship to it is crucial to such a highly self-conscious form of discourse. Not only does it influence the effectiveness and credibility of argumentation, but helps define what it means to write science.. . . [8] (p. 38)

We take as our guiding principle the thesis that these larger pragmatic functions of hedging indicate that other rhetorical strategies in scientific writing—for example, the use of citations—may be found to work together with hedges in creating persuasive effects and influencing interpersonal judgements.

1.3 Background to the Research

A *citation* may be formally defined as a portion of a sentence in a citing document which references another document or a set of other documents collectively. A *citation sentence* is any sentence in the full text body that contains at least one citation. A *citation window* corresponds to a citation sentence together with the preceding and following sentences, if they occur in the same paragraph.

Garzone and Mercer. Garzone and Mercer [4] motivated the current citation categorization project. This foundational work demonstrated a classifier system based on a correspondence between certain cue words, specific word usages, and characteristic structural patterns in citing sentences and the citation functions performed by the citing sentences. For example, in sentence 1.3, the phrase *still in progress* may be taken to indicate that the citation is referring to work of a concurrent nature.

(1) Although the 3-D structure analysis by x-ray crystallography is <u>still in progress</u> (Eger et al., 1994; Kelly, 1994), it was shown by electron microscopy that XO consists of three submasses (Coughlan et al., 1986).

Di Marco and Mercer. Di Marco and Mercer [1] developed the first stages of a method for citation classification guided by the hypothesis that the fine-grained rhetorical structure of a scientific article can help tremendously in this task. This hypothesis is based on the following two arguments:

– The well-established body of work in rhetorical theory may be used in analyzing the global structure of scientific discourse, e.g., [3], [5], [6], [12].
– More-recent studies have demonstrated the role of fine-grained discourse cues [9] [10] in the rhetorical analysis of general text.

We are thus developing an approach to citation classification in which the recognition of such subtle linguistic cues, together with models of scientific argumentation, provide a means of constructing a systematic analysis of the role citations play in maintaining a network of rhetorical relationships among scientific documents. As a key part of our methodology, we intend to show that a direct mapping can be determined from the pragmatic functions of these linguistic cues to the rhetorical purpose of the citations in the context within which they are both used.

In our preliminary study [11], we analyzed the frequency of the cue phrases from [10] in a set of scholarly scientific articles. We reported strong evidence that these cue phrases are used in the citation sentences and the surrounding text with the same frequency as in the article as a whole. In subsequent work [1], we analyzed the same dataset of articles to begin to catalogue the fine-grained discourse cues that exist in citation contexts. This study confirmed that authors do indeed have a rich set of linguistic and non-linguistic methods to establish discourse cues in citation contexts.

We found that several types of syntactic stylistic usage provide rhetorical cues that may serve to indicate the nature of the citation. For example, the use of syntactic symmetry or parallelism can act as a cue for the enumeration of one or more citations. Repetition of words and phrases may also be considered a form of lexical 'parallelism' that occurs along with citations. Other forms of rhetorical cueing rely on various kinds of lexical stylistic features within a citation context: lexical morphology (e.g., contrasting concepts, such as *intra*cellular and *extra*cellular, that set up for a 'contrasting-work' citation); specific lexical choice (e.g., negative or 'extreme' words to describe results that seem to call for citations to supporting works); scientific procedural terms; and 'reporting' verbs (e.g., to make reference to the historical record of the author's own or other works). For this latter category of reporting cues, we observed the use of hedging verbs used along with a reporting style in citation contexts (e.g., *it has been previously suggested...* that led us to investigate the relationship between hedging and citation occurrences in more detail.

Teufel. Teufel [14] represents a direct contrast to the approach taken by Garzone and Mercer and, by extension, our own approach. Teufel's work is concerned with the automated generation of summaries of scientific articles using the rhetorical structure of the document to find specific types of information to fill slots in a 'fixed-form' summary template. Teufel proposes a detailed model of scientific argumentation that may be used as the basis for analyzing and summarizing the content of an article, including citation content. This model consists of 31 argumentative 'moves', which are typically one clause or sentence in length, and which build, step by step, the rhetorical structure of the scientific presentation.

Teufel diverges from us in questioning whether citations are in any way linguistically marked, and, in particular, whether fine-grained discourse cues even occur in citation contexts. Even if such "overt cues" do exist, she notes, the task of detection through automated means would be formidable, requiring either deep-linguistic analysis or use of only simple, short, well-edited texts. Teufel thus articulates the dual challenges facing us: to demonstrate that fine-grained linguistic cues can in fact play a role in citation analysis and that such cues can be detected by automated means.

Although Teufel's approach runs counter to ours in judging whether citation contexts may be classified on the basis of subtle linguistic markers, she does nonetheless give many instances of argumentative moves that may be signalled in citation contexts by specific cues. Of interest to us, Teufel's set of argumentative moves is reminiscent of the kinds of categories used in citation classification schemes, and, significantly, Teufel observes that an important assumption is that "the argumentative status of a certain move is visible on the surface by linguistic cues." (p. 84) However, Teufel voices her concern with the "potentially high level of subjectivity" (p. 92) inherent in judging the nature of citations. As a consequence, she relies on only two clearly distinguishable citation categories in developing her ultimate argumentative model : the cited work either provides a basis for the citing work or contrasts with it. In our approach, we hope to maintain both a very broad and varied set of citation categories, while at the same time developing methods for reliable citation classification based on the automated detection of subtle but pragmatically well-defined rhetorical cues.

2 The Frequency of Hedging Cues in Citation Contexts

The argumentative structure found in scientific writing is supported by a variety of rhetorical techniques, including hedging. Underlying our interest in studying citations in scientific writing is the supposition that citation use is a rhetorical strategy. The work reported here begins to investigate the rhetorical links between citation usage and rhetorical cues. The hypothesis that we test is that hedges play an important role in the rhetoric that is intrinsic to citation usage. We investigate this hypothesis by doing a frequency analysis of hedging in citation contexts in a corpus of scholarly biology articles.

We set out to compare the frequencies of hedging cues occurring in citation sentences and the sentences immediately surrounding the citation sentences to the frequency of hedging cues in the text as a whole. If these frequencies show a statistically significant difference, these differences may provide evidence for our hypothesis. A list of the hedging cues can be found in Appendix A. Writers make purposeful stylistic and rhetorical choices, therefore, we propose that frequency differences would be supporting evidence that hedging plays a role in the rhetoric that is intrinsic to citation usage. Demonstrating our hypothesis, in addition to providing further evidence to support Hyland's work, would give us reason to study the rhetorical relationship between hedging and citations at a much more detailed level.

3 Methodology

The corpus that was analyzed is a 985-document subset of the BioMed Central corpus[1] which is composed of biomedical journal articles. Only journal articles deemed to have a strong biological focus (as opposed to a medical focus) were included in the test corpus. All articles were required to have the following sections: background, methods, results, discussion, and conclusion. This structure

[1] http://www.biomedcentral.com/

is very similar to the IMRaD[2] structure used by Teufel [14]. Since many of the articles merged the results and discussion sections, the two were treated as a single, aggregate section.

The corpus was first preprocessed to remove any extraneous XML tags. Most of these tags are either located in the front matter, or are for the purposes of formatting. Only information about paragraph boundaries, citation locations, and section boundaries was kept. The following example illustrates a typical body sentence from the corpus. Only the <p> tag, which denotes the start of a paragraph, and the <abbr> tag, which denotes a citation were needed.

(2) <p>Previous studies have examined the nucleotide length distribution of the 5' UTRs, 3' UTRs, intergenic regions and space between RBSs and start sites of transcription in the genome of <it>E. coli </it><abbrgrp> <abbr bid="B5">5</abbr></abbrgrp>.

A major advantage of the BioMed Central corpus is that citations are explicitly marked through the use of the <abbr> tag. As such it was unnecessary to cope with the various possible citation styles. Furthermore, since we are only interested in the presence or absence of hedging cues in a sentence, no syntactic processing was required. Finally, no morphological processing was performed. Since the number of hedging cues is small, the list of hedging cues taken from Hyland's catalogue was simply expanded to include all inflectional variants. We did not search for all derivational variants, since there is some evidence that not all such variants are used for hedging purposes [8].

The corpus was then split into sentences using a maximum-entropy sentence boundary detector, called MXTERMINATOR[3], described in [13]. The model was trained on a manually segmented portion of the corpus, consisting of approximately 3000 sentences. Unfortunately, MXTERMINATOR did not perform well on the biological corpus, and significant manual post-processing was required to correct segmentation errors. It is not clear if this is a result of the small training-set size, or the increased complexity of biological articles as compared to the Wall Street Journal articles on which MXTERMINATOR was originally evaluated.

The manual post-processing consisted of searching the output text from MXTERMINATOR for indications that MXTERMINATOR likely made an error, then manually correcting the mistake. There were several such indicators, such as single-word sentences, capital letters finishing a sentence (problematic since terms like "E. coli" were often split across sentences), and citations located at the beginning of a sentence (since these should be associated with the previous sentence, not the subsequent one), and so on.

Once segmentation was complete, each sentence in the corpus was identified as one or more of the following:

– A citation sentence, if the sentence contains one or more citations.

[2] Introduction, Methods, Results, and Discussion
[3] http://www.cis.upenn.edu/~adwait/statnlp.html

– A citation frame sentence, if the sentence contains no citation and is immediately adjacent to a citation sentence that is within the same paragraph.
– A hedge sentence, if the sentence contains one or more hedging cues.

Citations are only counted if they are citations to published work. Hence, citations of unpublished data, personal communications and so on are not counted as citations for our purposes. Such citations to unpublished sources are not marked up in the BioMed corpus, so no additional processing was needed to exclude them.

On occasion a citation may occur immediately following the period at the end of a sentence. In such instances, the citation is included as part of the immediately preceding sentence, rather than the following sentence. The following example illustrates such a case.

(3) Studies have shown highest apoptosis expression in lining epithelium at estrus in mouse <citation/> and rat. <citation/><citation/>

Several tallies were computed. We kept track of each citation sentence and frame, noting whether each contained a hedging cue. In addition, each citation window, which comprises both the citation sentence and the citation frame, was noted as either containing or lacking a hedging cue. Finally, we tallied the total number of sentences that contain a hedging cue, the total number of sentences that contain a citation, and the total number of sentences that fall into a citation frame.

It was often the case that citation windows overlapped in the text. This is especially evident in the citation-rich background section. When this occurred, care was taken to avoid double-counting hedging cues. When a hedging cue occurred in the intersecting region of two citation windows, the cue was counted as belonging to only one of the two windows. If it was in the citation sentence of one of the two windows, it was counted as belonging to the citation sentence in which it fell. If it fell in the intersection of two citation frames, it was counted as belonging to the citation that had no other hedge within its window. If neither window contained any other hedging cues, it was arbitrarily treated as belonging to the first of the two windows.

4 Results and Discussion

Table 1 shows the counts and Table 2 shows the frequencies of citation sentences, frame sentences, and hedge sentences. Any given sentence may belong to only one of the citation/frame categories. Since citation windows may overlap, it is sometimes the case that a citation sentence may also be part of the frame of another window. In this case, the sentence is counted only once, as a citation sentence, and not as a citation-frame sentence. Note that in Table 2, the frequencies do not add to 1, since there are sentences that neither occur in a citation window nor contain hedging cues. Data about these sentences has not been listed in Table 2.

Frame Sentence	<p>To test this idea further, we also analyzed a construct where the third Val residue in the V18 segment was changed to Pro.
Citation Sentence	We have previously shown that the introduction of a Pro residue in corresponding positions in a L23V transmembrane segment leads to a reduction in the MGD value of about 2.5 residues, presumably as a result of a break in the poly-Leu -helix caused by the Pro residue [<citation/>14].
Frame Sentence	Indeed, the initial drop in the glycosylation profile for the V18(P3) construct was $\sim$ 2 residues, Fig. 4B, while the shift in the location of the second drop was only $\sim$ 1 residue.
Normal Sentence	This is consistent with the possibility that V18 molecules with MGD $\sim$ 15.5 residues indeed have already formed a transmembrane-helix at the time of glycosylation, whereas the remaining ones have not.

Fig. 1. A paragraph containing all three sentence types. There are two hedge cues (underlined) in this example, one in the citation frame, and one outside the citation window.

Hedge sentences are further subdivided into verb and non-verb categories depending on whether the hedging cue is a verb or a non-verb. Note that a sentence may belong to both of these categories. The reason for this is that the sentence may contain two cues, one from each category. In all cases, a sentence containing more than one hedging cue is counted only once as a hedge sentence (reported in the 'Total' column). This single-counting of sentences containing multiple cues explains why the number of hedge sentences do not add up to the total number of hedging cues.

Table 3 shows the proportions of the various types of sentences that contain hedging cues, broken down by hedging-cue category. For all but two combinations, citation sentences are more likely to contain hedging cues than would be expected from the overall frequency of hedge sentences ($p \leq .01$). The two combinations for which there are no significant differences are non-verb hedging cues in the background and conclusion sections. It is interesting to note that there are, however, significantly ($p \leq .01$) more non-verb cues than expected in citation frames in the conclusion section.

With the exception of the above combination (non-verb cues in the conclusion section), citation frame sentences seem to contain approximately the same proportion of hedging cues as the overall text. However, this being said, there is little indication that they contain fewer cues than expected. The one major exception to this trend is that citation frame sentences in the background section appear less likely to contain verbal hedging cues than would be expected. It is not clear whether this is due to an actual lack of cues, or is simply an artifact of the fact that since the background section is so citation rich, there are relatively few citation frames counted (since a sentence is never counted as both a citation sentence and a citation frame sentence).

The $\chi^2(1, n)$ values for observed versus expected proportion of citation sentences and frame sentences containing hedging cues are summarized in Table 4.

Table 1. Number of sentences, by sentence type.

	Total	Citation		Hedge Sentences		
	Sentences	Sentences	Frames	Verb	Non-verb	Total
background	22321	10172	6037	2891	2785	5278
methods	36632	5922	5585	2132	1480	3468
res+disc	87382	16576	16405	13602	12040	23198
conclusions	5145	587	647	1049	760	1635

Table 2. Proportion of total sentences, by sentence type.

	Citation		Hedge Sentences		
	Sentences	Frames	Verb	Non-verb	Total
background	0.46	0.27	0.13	0.12	0.24
methods	0.16	0.15	0.06	0.04	0.09
res+disc	0.19	0.19	0.16	0.14	0.27
conclusions	0.11	0.13	0.20	0.15	0.32

$\chi^2(1, n)$ values were computed by comparing the actual versus expected frequencies of hedging cues in each sentence type. The expected frequencies are obtained simply from the overall frequency of each sentence type. Thus, if hedging cues were distributed randomly, and 24% of sentences overall had hedging cues, one would expect that approximately 24% of citation sentences would contain cues, assuming there is no relationship between hedging and citations. In order to correct for multiple χ^2 tests, Bonferroni correction was applied.

Tables 5, 6, and 7 summarize the occurrence of hedging cues in citation windows. Table 8 shows the proportion of hedge sentences that either contain a citation, or fall within a citation frame. Note that this is not the same thing as the proportion of *hedging cues* that fall within a citation sentence or frame. If more than one hedging cue falls within a single sentence, the sentence is counted as a single hedge sentence.

Table 8 suggests (last 3-column column) that the proportion of hedge sentences containing citations or being part of citation frame is at least as great as what would be expected just by the distribution of citation sentences and citation windows. Table 3 indicates that in most cases the proportion of hedge sentences in the citation windows is greater than what would be expected by the distribution of hedge sentences. Taken together, these conditional probabilities support the conjecture that hedging cues and citation contexts correlate strongly. Rather than occurring by chance, writers purposefully use these cues. With this knowledge, the strong correlation would indicate that the hedging cues are being used in synergy with the citation contexts. Hyland has catalogued a variety of pragmatic uses of hedging cues, so it is reasonable to speculate that these uses map over to the rhetorical structure that is found in citation contexts.

Table 3. Proportion of sentences containing hedging cues, by type of sentence and hedging cue category.

	Verb Cues			Non-verb Cues			All Cues		
	Cite	Frame	All	Cite	Frame	All	Cite	Frame	All
background	0.15	0.11	0.13	0.13	0.13	0.12	0.25	0.22	0.24
methods	0.09	0.06	0.06	0.05	0.04	0.04	0.14	0.10	0.09
res+disc	0.22	0.16	0.16	0.15	0.14	0.14	0.32	0.27	0.27
conclusions	0.29	0.22	0.20	0.18	0.19	0.15	0.42	0.36	0.32

Table 4. $\chi^2(1, n)$ values for observed versus expected proportion of citation sentences and frames containing hedging cues. $\chi^2_{crit} = 9.14$ after Bonferroni correction.

	n		Verb Cues		Non-verb Cues		All Cues	
	citation	frame	citation	frame	citation	frame	citation	frame
background	10172	6037	32.66	22.19	0.97	0.93	15.69	5.65
methods	5922	5585	118.75	0.94	13.53	0.03	113.82	1.33
res+disc	16576	16405	451.48	0.58	20.53	2.01	288.36	4.19
conclusions	587	647	24.50	1.17	5.57	9.92	26.86	6.16

5 Conclusions and Future Work

In creating inter-article relationships, citations in scientific writing reinforce the argumentation structure that is intrinsic to all scientific writing. To determine the relationship between a citing and cited paper, we are unravelling the argumentation structure by looking for fine-grained discourse and rhetorical cues that indicate the rhetorical relations that build the argumentation structure. One type of rhetorical cue involves hedging to modify the affect of a scientific claim. In this paper we show that the hedging cues proposed in Hyland's extensive study of the rhetorical use of hedging in scientific writing occur more frequently in citation contexts than in the text as a whole. We have also confirmed that hedging is distributed unevenly in the different sections of a scientific article. With this information we conjecture that hedging cues are an important aspect of the rhetorical relations found in citation contexts and that the pragmatics of hedges may help in determining the purpose of citations.

The pragmatic nature of hedges can be divided into several categories, most broadly, *content-oriented hedges* and *reader-oriented hedges*. Content-oriented hedges "hedge the correspondence between what the writer says about the world and what the world is thought to be like" [8] (p. 162). Reader-oriented hedges are concerned with the social relationship between writer and reader, and are used for such pragmatic purposes as reducing the potential risks of presenting controversial claims, showing courtesy or deference to peers, and demonstrating conformity to the research community's expectations [8] (pp. 177-178). In the following example, the use of a personal subject (*We*) functions to mitigate the

Table 5. Number and proportion of citation windows containing a hedging cue, by section and location of hedging cue.

	Windows		Sentences		Frames	
	#	%	#	%	#	%
background	3361	0.33	2575	0.25	2679	0.26
methods	1089	0.18	801	0.14	545	0.09
res+disc	7257	0.44	5366	0.32	4660	0.28
conclusions	338	0.58	245	0.42	221	0.38

Table 6. Proportion of citation windows containing a verbal hedging cue, by section and location of hedging cue.

	Windows		Sentences		Frames	
	#	%	#	%	#	%
background	1967	0.19	1511	0.15	1479	0.15
methods	726	0.12	541	0.09	369	0.06
res+disc	4858	0.29	3572	0.22	2881	0.17
conclusions	227	0.39	168	0.29	139	0.24

criticism of other work through an "overt acceptance of personal responsibility" [8] (p. 181). In addition, the modal *might* further weakens the critical effect:

(4) We do not know the reason for the discrepancy between our results and those of Ngernprairitsiri [16, 23], but it might reflect genetic differences in the cultivars employed.

As this last example shows, the pragmatic purpose of hedges and citations occurring within the same passage can be closely linked. As we observed in our study, specific types of hedges were associated with citations that offset the implied uncertainty. One type of content-oriented hedge expresses deviation from established or idealized knowledge [8] (p. 164) and is often associated with a citation to foundational work. In the following example, the adverb *slightly* indicates that the procedure being used deviates from that of Buchner *et al.*:

(5) Drosophila heads were harvested and prepared according to a slightly modified protocol described in Buchner et al. [38].

We are now beginning to develop a mapping from the pragmatic functions of hedging cues to the purpose of citations used within the same context. Our ultimate purpose is to identify fine-grained linguistic cues that may be used as a means of determining the function of citations. Based on Hyland and others, we can expect to be able to associate hedging cues and other cue phrases with rhetorical relations as determiners of citation function.

Acknowledgements. Our research has been financially supported by the Natural Sciences and Engineering Research Council of Canada and by the Univer-

Table 7. Proportion of citation windows containing a non-verb hedging cue, by section and location of hedging cue.

	Windows		Sentences		Frames	
	#	%	#	%	#	%
background	1862	0.18	1302	0.13	1486	0.15
methods	432	0.07	295	0.05	198	0.03
res+disc	3751	0.23	2484	0.15	2353	0.14
conclusions	186	0.32	107	0.18	111	0.19

Table 8. Proportion of hedge sentences that contain citations or are part of a citation frame, by section and hedging cue category.

	Verb Cues			Non-verb Cues			All Cues		
	Cite	Frame	None	Cite	Frame	None	Cite	Frame	None
background	0.52	0.23	0.25	0.47	0.28	0.25	0.49	0.26	0.26
methods	0.25	0.16	0.59	0.20	0.15	0.65	0.23	0.16	0.61
res+disc	0.26	0.19	0.55	0.21	0.19	0.60	0.23	0.19	0.58
conclusions	0.16	0.14	0.70	0.14	0.16	0.70	0.15	0.14	0.71

sities of Western Ontario and Waterloo. Thanks also goes to Erin Harvey and Heather Thiessen in the Department of Statistics and Actuarial Science, University of Waterloo, for assistance with statistical design and analysis.

References

1. Di Marco, C. and R. E. Mercer: Toward a catalogue of citation-related rhetorical cues in scientific texts. In *Proceedings of the Pacific Association for Computational Linguistics Conference* (PACLING'03) (2003) 63–72
2. Di Marco, C. and R. E. Mercer: Hedging in scientific articles as a means of classifying citations. To appear in *AAAI Spring Symposium on Exploring Attitude and Affect in Text* (2004)
3. Fahnestock, J.: *Rhetorical figures in science.* Oxford University Press (1999)
4. Garzone, M. and R. E. Mercer: Towards an automated citation classifier. In *AI'2000, Proceedings of the 13th Biennial Conference of the CSCSI/SCEIO*, Lecture Notes in Artificial Intelligence, v. 1822, H. J. Hamilton (ed.), Springer-Verlag, (2000) 337–346
5. Gross, A.G.: *The rhetoric of science.* Harvard University Press (1996)
6. Gross, A.G., J. E. Harmon, and M. Reidy: *Communicating science: The scientific article from the 17th century to the present.* Oxford University Press (2002)
7. Halliday, M. A. K.: Functional diversity in language as seen from a consideration of modality and mood in English. *Foundations of Language*, Volume 6, (1970) 322–361
8. Hyland, K.: *Hedging in scientific research articles.* John Benjamins Publishing Company (1998)

9. Knott, A.: *A data-driven methodology for motivating a set of coherence relations.* Ph.D. thesis, University of Edinburgh (1996)
10. Marcu, D.: *The rhetorical parsing, summarization, and generation of natural language texts.* Ph.D. thesis, University of Toronto (1997)
11. Mercer, R. E. and C. Di Marco: The importance of fine-grained cue phrases in scientific citations. In *Proceedings of the 16th Conference of the CSCSI/SCEIO* (AI'2003) (2003) 550–556
12. Myers, G.: *Writing biology.* University of Wisconsin Press (1991)
13. Reynar, J. C. and A. Ratnaparkhi: A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Washington, D. C. (1997)
14. Teufel, S.: *Argumentative zoning: Information extraction from scientific articles.* Ph.D. thesis, University of Edinburgh (1999)

A Hedging Cues

Table 9. Base forms of all verb hedging cues used in the analysis.

appear	calculate	indicate	report	speculate
assume	estimate	note	see	suggest
attempt	imply	predict	seek	suspect
believe	indicate	propose	seem	

Table 10. All non-verb hedging cues used in the analysis.

about	essentially	partial	rarely
almost	evidently	partially	relatively
apparent	generally	possibility	slightly
apparently	likely	potentially	some
approximate	most	presumably	somewhat
approximately	mostly	probable	unlikely
around	normally	probably	usually
consistent	occasionally	quite	virtually

Each verbal cue given in Table 9 was expanded into four inflectional variants in the analysis presented in this paper: the base form, the third person singular present tense, the past tense, and the present participle form (e.g. appear, appears, appeared, and appearing, respectively).

Finding Interesting Summaries in GenSpace Graphs Efficiently

Liqiang Geng and Howard J. Hamilton

Department of Computer Science
University of Regina, Regina, SK, S4S 0A2
{gengl, Hamilton}@cs.uregina.ca

Abstract. Summarization based on GenSpace graphs aggregates data into summaries in many ways and identifies summaries that are far from user expectations. Mining interesting summaries in GenSpace graphs involves expectation propagation and interestingness measure calculation in the graphs. Both the propagation and the calculation need to traverse the GenSpace graph, but the number of the nodes in the GenSpace graph is exponential in the number of attributes. In this paper, we propose pruning methods in the different steps of the mining process: pruning nodes in ExGen graphs before constructing the GenSpace, pruning nodes in GenSpaces before propagation, and pruning nodes in GenSpaces during propagation. With these methods we make the traverse more efficient, by reducing the number of the nodes visited and the number of records scanned in the nodes. We also present experimental results on the Saskatchewan weather data set.

1 Introduction

Summarization, which is the formation of interesting, compact descriptions of data, was identified by Fayyad et al. as one of the six chief data mining tasks [3]. Early work on attribute-oriented induction [2] and function finding [12] provided diverse approaches to summarization. Summarization is the crucial task addressed by online analytical processing (OLAP), data cubes with rollup and drilldown operators, and the *rollup* and *cube by* operators in SQL. Recent research has focused on the expansion of the functions for exploiting the data cubes and the improvement on efficiency. The use of iceberg cube has been proposed to selectively aggregate summaries with an aggregate value above minimum support thresholds [1]. This approach assumes that only the summaries with high aggregate values are interesting, which is not always the case. Sarawagi et al. introduced an exploitation tool to assist the user to find exceptions in data cubes based on statistical models [9]. A `diff` operator was proposed to simplify the process by automatically finding the underlying reasons for an exception [8].

Nonetheless, these approaches to summarization have three weaknesses. First, many valid summaries of the same data or its subsets can be produced, and during exploration, the data analyst must examine summaries one by one to assess them. Secondly, the capacity for incorporating existing knowledge is limited. Although attribute-oriented induction allows the incorporation of domain knowledge as a concept hierarchy for each attribute, multiple ways of aggregating the values for an attribute during a single data mining task are not permitted. Similarly, hierarchical statistical models allow expectations, in the form of a priori distributions, to be specified, but only for a single hierarchy of distributions. Thirdly, the ability to respond to changes in the user's knowledge during the knowledge discovery process is limited. To overcome these limitations, we proposed a method of finding interesting summaries based on users' expectations by incorporating the generalization information and users' expectations in ***generalization space*** (***GenSpace***) graphs [6].

A ***domain generalization graph*** (DGG) [6,10] is a graphical structure that denotes a concept hierarchy for an attribute. The nodes in a DGG are domains of values, and the arcs tell how to generalize values from one domain to values in another. Each path in a DGG corresponds to a generalization consistent with the specified generalization relations. An ***expected distribution domain generalization graph*, or *ExGen graph***, is a DGG where each node has been augmented with a description of the expected distribution (or simply ***expectations***) of the values in the corresponding domain (Here the expectation means an estimated probability of occurrences of a value for a random variable, therefore, it is different from a mathematical expectation in statistics, which represents the mean value of a random variable). Initial and changing domain knowledge relevant to summarization of a single attribute is readily described by ExGen graphs. During the knowledge discovery process, the expectations at a particular node, which reflect a user's knowledge about the corresponding domain, can be updated. As well, expectations can be propagated to other nodes in the graph; for example, if the user revises the expectation about the number of shows watched in the evening, then the expectation about the number of shows watched from 8:00PM to 9:00PM can be automatically adjusted.

For multiple attributes, we can obtain a GenSpace graph from the ExGen graphs for the individual attributes. In a GenSpace graph, each node denotes a summarization of the underlying data table. Each arc tells how to generalize values from one domain to values in another for a single attribute.

The GenSpace summary mining process has five steps. First, a domain generalization graph (DGG) for each attribute is created by explicitly identifying the domains appropriate to the relevant levels of granularity and the mappings between the values in these domains. The expectations (expected probability distributions) are specified by the user for some nodes in each DGG to form an ExGen graph and then propagated to all the other nodes in the ExGen graph. Second, the framework of the GenSpace graph is generated based on the ExGen graphs for individual attributes, and the potentially interesting nodes in this graph are materialized. Third, aggregation is performed by transforming values in one domain to another, according to the directed arcs in the DGG graphs. Fourth, the given expectations are propagated throughout the GenSpace subgraph consisting of potentially interesting nodes, and the interestingness measures for these nodes are calculated. Fifth, the highest ranked summaries are displayed. Expectations in the GenSpace graph are then adjusted and steps are repeated as necessary. In step four, the expectation propagation and the interestingness calculation involve traversing the GenSpace graph and scanning the records in the nodes. The cost of traversing and scanning is a key factor that influences the efficiency of the system. In this paper, we address efficiency issues related to the propagations in different stages.

The remainder of this paper is organized as follows. In the following section, we give background of the GenSpace summary mining. In Section 3, we propose the creation of virtual bottom nodes for ExGen graphs before the GenSpace graph is created. In Section 4, we propose a method for pruning uninteresting nodes before propagation. In Section 5, we propose interestingness measures and a pruning strategy to reduce the number of the nodes during the propagation process. In Section 6, we present experimental results on Saskatchewan weather data. Finally, in Section 7, we present our conclusions.

2 GenSpace Graphs and Propagation of Expectations

First, we give formal definitions for ExGen and GenSpace graphs.

Definition 1. (adapted from [10]) Given a set $X = \{x_1, x_2, \ldots, x_n\}$ representing the domain of some attribute and a set $P = \{P_1, P_2, \ldots, P_m\}$ of partitions of the set X, we define a nonempty binary relation $\preceq$ (called a ***generalization relation***) on P, where we say $P_i \preceq P_j$ if for every section $S_a \in P_i$, there exists a section $S_b \in P_j$, such that $S_a \subseteq S_b$. A ***domain generalization graph*** (*DGG*) $G = \langle P, Arc \rangle$ is constructed based on a generalization relation $\langle P, \preceq \rangle$ as follows. The nodes of the graph are the elements of P. There is a directed arc from P_i to P_j in *Arc* iff $P_i \neq P_j$, $P_i \preceq P_j$, and there is no $P_k \in P$ such that $P_i \preceq P_k$ and $P_k \preceq P_j$. Each node corresponds to a domain of values called ***sections***. Each arc $a \in Arc$ corresponds to a generalization relation, which is a mapping from the values in the domain of the initial (or ***parent***) node to that of the final node (or ***child***) of the arc. The ***bottom*** (or ***source***) node of the graph corresponds to the original domain of values X and the ***top*** (or ***sink***) node T corresponds to the most general domain of values, which contains only the value *ANY*.

In the rest of the paper, we use "summary", "node", and "partition" interchangeably.

Example 1. Let *MMDDMAN* be a domain of morning, afternoon, and night of a specific non-leap year {*Morning of January* 1, *Afternoon of January* 1, *Night of January* 1, …, *Night of December* 31}, and *P* a set of partitions {*MMDD*, *MM*, *Week*, *MAN*}, where *MMDD* = {*January* 1, *January* 2, …, *December* 31}, *MM = {January*, *February*, …, *December*}, *Week* = {*Sunday*, *Monday*, …, *Saturday*}, and *MAN* = {*Morning*, *Afternoon*, *Night*}. Here *MMDD* $\preceq$ *MM* and *MMDD* $\preceq$ *Week*. We obtain the DGG shown in Figure 1.

Definition 2 (**Expected distribution domain generalization graph**) An ***expected distribution domain generalization*** (or ***ExGen***) ***graph*** $\langle P, Arc, E \rangle$ is a DGG that has an expectation associated with every node. Expectations represent the expected probability distribution of occurrence of the values in the domain corresponding to the node. For a node (i.e., partition) $P_j = \{S_1, \ldots, S_k\}$, we have $\forall S_i \in P_j,\ 0 \le E(S_i) \le 1$ and $\sum_{i=1}^{k} E(S_i) = 1$, where $E(S_i)$ denotes the expectation of occurrence of section S_i.

Continuing Example 1, we assign expectations to each node in the DGG. For example, for the partition *MAN* = {*Morning*, *Afternoon*, *Night*}, we associate the expectations [0.2, 0.5, 0.3], i.e., *E*(*Morning*) = 0.2, *E*(*Afternoon*) = 0.5, and *E*(*Night*) = 0.3. We then obtain an ExGen graph.

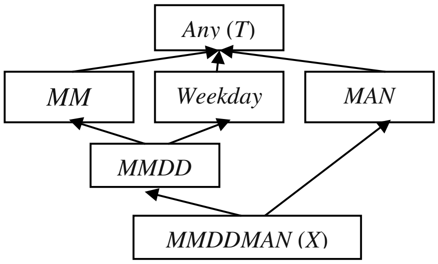


Fig. 1. An Example DGG

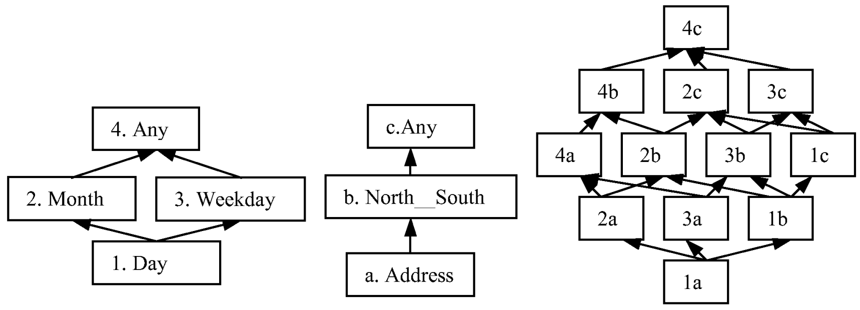
An ExGen graph is based on one attribute. If the generalization relation is defined on a set of attributes, we can construct a GenSpace graph to represent the complete space of possible generalization.

Definition 3. Given a set of attributes $\{A_1, A_2, \ldots A_n\}$, and an ExGen graph $G_i = <P_i, Arc_i, E_i>$ for each attribute A_i, a generalization space is defined as a triplet $<P, Arc, E>$. The set of nodes is $P = P_1 \times P_2 \times \ldots \times P_n$. For two nodes $Q = [P_{Q1}, P_{Q2}, \ldots, P_{Qn}] \in P$ and $R = [P_{R1}, P_{R2}, \ldots, P_{Rn}] \in P$, where $P_{Qi} \in P_i$ and $P_{Ri} \in P_i$ denote the partition of attribute A_i in nodes Q and R, respectively, if $P_{Qi} \preceq P_{Ri}$ for $1 \le i \le n$, we say $Q \preceq R$. There is a directed arc from Q to R in Arc iff $Q \neq R$, $Q \preceq R$, and there is no $O \in P$ such that $Q \preceq O$ and $O \preceq R$. The generalization relationship $\preceq$ is a partial order relation and $\langle P, \preceq \rangle$ defines a partially ordered set. For each node, we attach a set of expectations (expected probability distribution of occurrence) for the values in the domain corresponding to the node. For a node (i.e., partition) $Q = \{S_1, \ldots, S_k\}$, we have $\forall S_i \in Q, 0 \le E(S_i) \le 1$ and $\sum_{i=1}^{k} E(S_i) = 1$, where $E(S_i)$ denotes the expectation of occurrence of section S_i. A graph constructed in this way is called a ***generalization space graph***, or ***GenSpace graph***, $\langle P, Arc, E \rangle$.

The top node (or sink node) corresponds to the most general domain of values, which consists of only the tuple [*ANY*, *ANY*, …, *ANY*].

Example 2. Figure 2 illustrates the construction of a GenSpace graph. Given the two ExGen graphs (expectations not shown) representing the generalization relations for attributes *Day* and *Address* in Figure 2(a) and 2(b), we generate the GenSpace graph in Figure 2(c). Each node in the GenSpace represent a combination of the corresponding nodes in the ExGen graphs, and the expectations in a node represent the joint probability distribution for the corresponding attributes.

In the process of mining interesting summaries in the GenSpace graph, when the user specifies the expectations at a certain node, the expectations are propagated to the bottom node, and from there, they are propagated to the entire graph. The advantage of this propagation method is that it preserves consistency among the expectations at all nodes as proven in [4]. Sections 3, 4 and 5 address the efficiency issues for the bottom up propagation.



(a) ExGen Graph for *Day* (b) ExGen Graph for *Address* (c) GenSpace Graph

Fig. 2. Costructing a GenSpace graph for *Day* and *Address*.

3 Constructing ExGen Graphs with Virtual Bottom Nodes

Since an ExGen (or DGG) graph is used to represent the ontology for a domain, it can be fairly complex. For example, in [10] a calendar DGG with 27 nodes is proposed. However, in a specific application, perhaps only a small subset of the nodes is interesting to a user. If the user can mark uninteresting nodes in ExGen graphs for individual attributes, the ExGen graphs can be pruned before the GenSpace graph is created, and the resulting GenSpace will be smaller than it would have been. However, if the bottom node of an ExGen graph is marked as uninteresting, pruning the bottom node will prevent propagation from being performed consistently, because we convert all other types of propagation into bottom up propagation and we use the bottom node as the basis for this propagation. Preserving the structure of GenSpace graph with a single bottom node allows us to preserve the consistency of propagation. On the other hand, bottom nodes are usually very big and the bottleneck of storage and propagation. Therefore, we introduce a substitute bottom node or ***virtual bottom node*** that has a smaller size than the real bottom node but still has a generalization relation with all other interesting nodes in the ExGen graph.

Definition 4 (**Base Interesting Node**). In an ExGen graph G, let the set of interesting nodes be I. A node B is a ***base interesting node*** iff $B \in I$ and there does not exist an interesting node $A \in I$ such that $A \preceq B$.

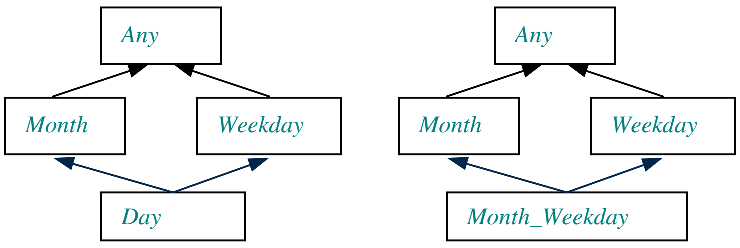
Definition 5 (**Virtual Bottom Node**). The ***virtual bottom*** node V of ExGen graph G is the combination of the partitions of base interesting nodes B_i, such that V satisfies $X \preceq V$ and $V \preceq B_i$, and there does not exist a partition V' such that $X \preceq V'$, $V' \preceq B_i$, and $V \preceq V'$.

If there is only one base interesting node B, the virtual bottom node V is identical to B. V is a rough partition of the bottom node X and the roughest possible fine partition of every base interesting node B_i. In the worst case, $V = X$.

Here we seek indiscernible relations based on different levels of generalizations. Yao et al. proposed a granulation based on indiscernible relations on multiple attributes [11], while our indiscernible relations are based on one attribute, but multiple generalizations of that attribute. Figure 3 gives our algorithm that constructs a virtual bottom node for an ExGen graph. The worst case complexity for this algorithm is $O(n(m + \log n))$, where n is the cardinality of the bottom node *BT* and m is the cardinality of *BI*.

```
Algorithm ConstructVB
Input: attribute a, its corresponding ExGen graph EG, the set of base interesting
nodes BI, bottom node table BT.
Output: virtual bottom node VB.
VB = {};
If there is only one node b in BI,
    Return VB = b;
For each value bvalue in BT,
    VirtualValue = "";
    For each base interesting node $b \in BI$ do
        // Find the generalized value of bvalue in b.
        GenValue = FindGenValue (bvalue, b);
        // Concatenate the generalized value to the current virtual value.
        VirtualValue = Concatenate(VirtualValue, GenValue);
    If VirtualValue is not in VB
        Insert VirtualValue into VB;
```

Fig. 3. Algorithm to Construct a Virtual Bottom Node.



(a) Original ExGen Graph (b) ExGen Graph with a Virtual Bottom Node

Fig. 4. Virtual Bottom Node.

It can be seen that propagation in the revised GenSpace graph has the same effect as in the original one.

Example 3. Figure 4 illustrates the definition of a virtual bottom node. Figure 4(a) is an ExGen graph for a *Date* attribute for a specific non-leap year. Suppose that only the bottom node is specified as uninteresting and that all possible values occur in the data set. We have a domain of 365 elements in the bottom node *Day*, 12 sections for node *Month* and 7 sections for node *Weekday*. The base interesting nodes are *Month* and *Wekday*. Figure 4(b) is the corresponding ExGen graph with a virtual bottom node *Month_Weekday*, with 12 * 7 = 84 sections {*January Sunday*, *January Monday*, …, *December Saturday*}. We can see that node *Month_Weekday* is a finer partition of the *Month* and *Weekday* nodes, and is a rough partition of the *Day* node.

4 Pruning Uninteresting Nodes from GenSpace Graphs

After the framework of a GenSpace graph has been generated, the user can mark some nodes as uninteresting before the propagation process starts. Usually, the uninteresting nodes are in the lower levels of the GenSpace graph, because these nodes correspond to larger summary tables, which are harder to comprehend. Pruning even a small percentage of such nodes will significantly reduce storage costs. We can prune an uninteresting node by connecting all its parent nodes to its child nodes, but this will destroy the one-step-generalization feature of a link. To preserve this feature, some uninteresting nodes must not be pruned. Our problem is to find the uninteresting nodes that should be preserved.

Example 4. In Figure 5, the white ovals (nodes N_2, N_3, N_4, and N_5) denote the uninteresting nodes and the black ovals denote the potentially interesting nodes. The numbers in parentheses denote the space cost of the nodes. We intend to prune a subset of uninteresting nodes such that all potentially interesting nodes can be reached from the bottom node. We can prune N_3 and N_5, or N_2 and N_5, or N_2 and N_4 to satisfy this constraint. Here we prefer to prune N_2 and N_4, because these two nodes require the most storage units (100 + 300 = 400).

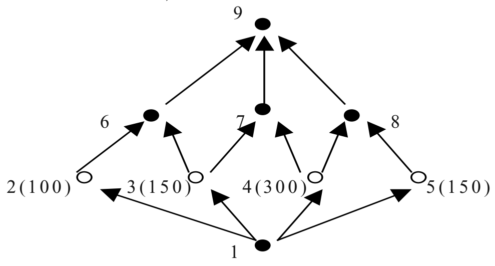


Fig. 5. Example 4.

Before we give the formal description of the problem, we first give some definitions.

Definition 6. If a non-uninteresting node only has uninteresting nodes as its parents, we call it an ***endangered node***.

If pruning is not properly performed, these nodes might not obtain expectations from bottom up propagations.

Definition 7. The parent nodes of endangered nodes are called ***candidate nodes***.

We need to select a subset of the candidate nodes to prune and preserve the rest in the GenSpace graph, although we will hide them from the user.

Definition 8. A candidate node's endangered children are called the ***cover of the candidate node***.

The ***node preservation problem*** is to find a subset of the candidate nodes such that the union of their covers equals the set of endangered nodes and the storage cost of these nodes is minimum.

Example 5. In Figure 5, the endangered node set is $\{N_6, N_7, N_8\}$, the candidate node set is $\{N_2, N_3, N_4, N_5\}$. $\text{Cover}_{N2} = \{N_6\}$, $\text{Cover}_{N3} = \{N_6, N_7\}$, $\text{Cover}_{N4} = \{N_7, N_8\}$, and $\text{Cover}_{N5} = \{N_8\}$. We represent these facts in Table 1.

Table 1. Cover Sets.

	N_6	N_7	N_8
N_2	100		
N_3	150	150	
N_4		300	300
N_5			150

In Table 1, each row describes a candidate node, and each column describes an endangered node. If a candidate is a parent of an endangered node, we put the storage cost in the corresponding cell. Here $\{N_3, N_5\}$ is a subset that covers the entire set of endangered nodes, and its storage cost (150 + 150 = 300) is minimum. Therefore, we choose to keep N_3 and N_5 and prune nodes N_2 and N_4.

We use a greedy heuristic to obtain an approximate solution to the node preservation problem. Since we want to choose the nodes that have smaller storage and greater coverage, we define the storage coverage ratio SCR(N) = Storage (N) / Coverage (N) for each candidate node N. At each step, we select a node with the lowest storage coverage ratio to preserve. After the selection, we remove the endangered nodes that are covered by the selected node, i.e., we delete their corresponding columns in the table. We also delete the row in the table corresponding to the selected node. Then we recalculate the storage coverage ratios in the new table and repeat the selection process. This process continues until all columns of the table are deleted, i.e., all the endangered nodes are covered. After obtaining the subset of nodes to preserve, we use backward elimination to eliminate any redundant nodes in this subset. The algorithm is presented in Figure 6.

Let m and n denote the numbers of the candidate and endangered nodes, respectively. The worst case occurs when only one more endangered node is covered each time we select a candidate node. If $m < n$, the worst case complexity is $O(m^2)$. If $m \geq n$, the worst case complexity for this algorithm $O(m(m - n))$.

In Example 5, because node N_3 has the minimum storage coverage ratio (150 / 2 = 75), we select it and eliminate nodes N_6 and N_7 of the endangered node set. Next, we choose N_5, because it has the minimum storage coverage ratio 150. (Initially, the storage coverage ratio of N_4 was 150, but after we eliminated N_7, the ratio became 300). At this point, all endangered nodes have been covered. So we keep N_3 and N_5 and prune N_2 and N_4.

After we select a new uninteresting node to preserve, it can become an endangered node again. We have to guarantee that all the newly selected candidate nodes are safe for propagation. Figure 7 gives the algorithm for selecting candidate nodes.

```
Function SelectCandidateNodesOneStep
1. Search the GenSpace graph and find all the endangered nodes.
2. Find the set of candidate nodes corresponding to the endangered nodes.
3. Construct the coverage relation table (as in Table 1).
4. While the set of the endangered node set is not empty,
    4.1 Select a candidate node with the minimum storage coverage ratio. If
    there is a tie, select one with greater coverage. If there is a tie again, we
    randomly select one.
    4.2 Eliminate the covered endangered nodes from the endangered node set
    to eliminate the selected candidate node from the candidate node set.
    4.3 Recalculate the coverage and storage coverage ratio for the left
    candidate nodes.
5. Check the selected node set one by one in the reverse order of selection
using forward adding backward elimination strategy and eliminate the
redundant nodes.
```

Fig. 6. Function SelectCandidateNodesOneStep

```
Algorithm SelectCandidateNodes
1.Create the set of endangered nodes by scanning the GenSpace graph.
2. While the set of endangered nodes with is not empty,
    2.1 Find the nodes to preserve using Function
    SelectCandidateNodesOneStep.
    2.2 Find the set of endangered nodes in the selected node set.
```

Fig. 7. Function SelectCandidateNodes

5 Pruning During the Propagation Process

After pruning nodes from the ExGen graphs and GenSpace graph, we begin the propagation and mining process. In this section, we prune nodes from the GenSpace in the propagation process using relative variance as an interestingness measure.

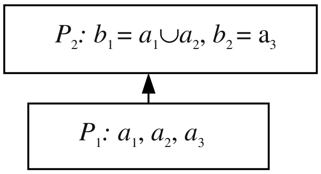
Intuitively, if the variance in a node is small enough, the variance of its descendent nodes will not be large. But this is not always true, because variance is not a monotonic function. In our system, we used the ***relative variance*** as an interestingness measure. We define the relative variance as $Intr\ (P) = \frac{1}{n}\sum_{i=1}^{n} \frac{|\, e(S_i) - o(S_i)\,|}{o(S_i)}$, where $e(S_i)$ and $o(S_i)$ are the expectations and observed probability of section S_i in node P. We define the ***relative difference*** for section S_i as $d(S_i) = \frac{|\, e(S_i) - o(S_i)\,|}{o(S_i)}$. We define $d(S_i) = 0$ if $o(S_i) = 0$ and $e(S_i) = 0$, and we define $d(S_i)$ to be the greatest value represented in the computer, if $o(S_i) = 0$ and $o(S_i) \neq e(S_i)$.

Relative variance was adopted due to two considerations. First, standard variance is biased towards the upper level nodes, because in the lower level nodes, where the number of the sections is larger than those in the upper level nodes, both the expectation and the observed probability for each section are very small, and the value of variance tends toward zero. Secondly, the relative variance has the following property, which can be used to prune nodes while searching for interesting summaries.

Theorem 1. In a GenSpace graph, assume that a node *P* consists of partition $\{S_1, S_2, \ldots, S_n\}$, and the relative variance are used as the interestingness measure. If for every partition S_i we have $d(S_i) = \frac{|e(S_i) - o(S_i)|}{o(S_i)} \le t, t \ge 0$, the relative variance of all descendents of *P* is less than *t*.

Theorem 1 shows that if all relative differences of every section in a node have an upper bound, then the interestingness measures of its descendents cannot exceed this upper bound.

Theorem 1 indicates that when we use the relative variance interestingness measure, the child's greatest relative difference is less than or equal to the parent's greatest relative difference, but it does not mean that the relative variance of the child must be less than that of the parent, as is demonstrated by the following example.



$o(a_1) = 0.3$, $e(a_1) = 0.4$
$o(a_2) = 0.3$, $e(a_2) = 0.4$
$o(a_3) = 0.4$, $e(a_3) = 0.2$
$o(b_1) = 0.6$, $e(b_1) = 0.8$
$o(b_2) = 0.4$, $e(b_2) = 0.2$

(a) GenSpace Graph

(b) Expectations and Observed Probability for Nodes P_1 and P_2

Fig. 8. An Example Illustrating Theorem 1.

Example 6. In Figure 8(a), node P_1 is a parent of node P_2. P_1 has three sections a_1, a_2 and a_3. P_2 has two sections b_1 and b_2, where b_1 is the union of a_1 and a_2, and b_2 is identical to a_3. The observations and expectations for the two nodes are listed in Figure 8(b). We obtain $d(a_1) = \frac{|e(a_1) - o(a_1)|}{o(a_1)} = 0.33$; similarly, we have $d(a_2) = 0.33$, $d(a_3) = 0.5$, $d(b_1) = 0.33$, $d(b_2) = 0.5$, $Intr(P_2) = 1 / 2 * (0.33 + 0.5) = 0.42$, $Intr(P_1) = 1 / 3 * (0.33 + 0.33 + 0.5) = 0.37$. Although the child's interestingness measure (0.42) is greater than that of its parent (0.37), it is not greater than the greatest relative difference of its parent node (0.5). If $t = 0.5$ has been set as the threshold, since all the differences of sections in P_1 is less than *t,* according to Theorem 1, the relative variances of the descendents of P_1 should be less than *t*, thus, the descendents of P_1 can be pruned.

Figure 9 illustrates a pruning strategy based on the relative variance measure. This search algorithm finds the *k* most interesting summaries. In a run, if the user accepted the expectations for the observed probability, i.e., the expectations and observed probabilities are identical, all its descendants will not be interesting and they will be pruned. This is the special case of our heuristic.

Unlike the pruning strategy proposed in Sections 3 and 4, this pruning method depends on the expectations specified by the user, as well as the GenSpace graph structure. Therefore, we concentrate on the theoretical aspects of this method, and will not report experimental results for this method in Section 6. Showing a few experimental results will not provide much insight into the general behavior.

In an implementation, the expectation propagation and interestingness measure calculation can be combined, because some nodes will be pruned based on interestingness and their distributions will not be needed for propagation.

```
Algorithm GSProp
Input: a GenSpace graph $G$, $k$ (find top $k$ nodes)
1. From bottom to top in $G$, select the first $k$ nodes using breadth first search
and mark them as the visited nodes.
2. Calculate the relative variances for the selected nodes.
3. Let $t$ be the smallest relative variance in the $k$ nodes.
4. For the next unvisited and unpruned node in the breadth first search,
        Calculate the difference between observations and expectations for
        each section.
        If the differences are all less than $t$,
                Prune this node and all its descendant nodes,
        otherwise,
                Calculate its interestingness measure.
                If the interestingness measure is greater than $t$,
                        Replace the node with the least relative variance in the
                        set of selected nodes with the new node.
                        Set $t$ to be the least relative variance of the currently
                        selected nodes.
```

Fig. 9. Algorithm GSProp.

6 Experimental Results

We implemented the aforementioned methods in our DGG-Discover 5.0 software. Due to the space limitation, we only present the results obtained on the Saskatchewan weather dataset.

In the weather dataset, the number of daily weather observations (tuples) was 211,534. We used the daily high temperature (in 0.1 degree Celcius) and daily total precipitation (in mm, with snow converted to equivalent water) for all weather stations for all days from January 1, 1900 to December 31, 1949. The experimental process and the results obtained were presented in [5]. In this paper, we concentrate on the efficiency issues.

The attributes we used in our experiments are *Station, Time*, *HighTemperature* (temperature in Celsius), and *TotalPrecip* (precipitation in mm). Attribute *Time* has format *YYYYMMDD*, including the information of year, month, and day. We generalize it to 6 nodes: *YYYYMM* (year and month), *MM* (month), *YYYY* (year), *Decade*, *Season* and *Any* (any time). *HighTemperature* was generalized to three nodes *TempRange*, *TempSplit*, and *Any*, *TotalPrecip* was generalized to three nodes *PrepRange*, *PrepSplit*, and *Any*, and *Station* was generalized to three nodes representing geographic features and node *Any*. DGG graphs for the four attributes are shown in Figure 10. In the GenSpace graph, there are (6 + 1) * (3 + 1) * (3 + 1) * (4 + 1) = 560 nodes (559 nodes corresponding to summaries plus the bottom node corresponding to the raw data).

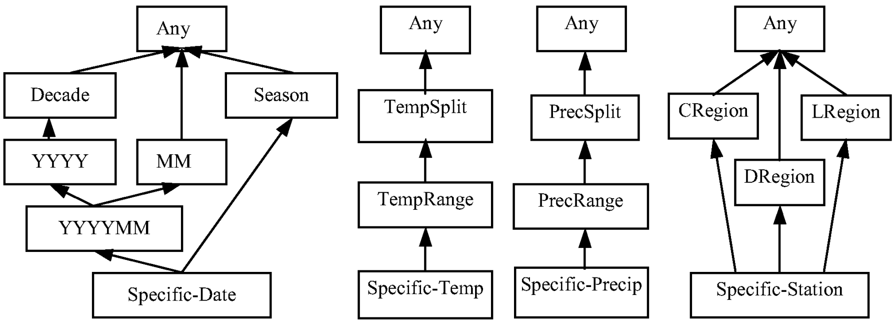


Fig. 10. DGGs for *Date*, *HighTemperature*, *TotalPrecip*, and *Station* Attributes

Harinarayan proposed a linear time cost model for summarizing a table [7]. They found that the time cost for a summarization is directly proportional to the size of the raw table. Therefore, in our case, when we summarize node *B* from node *A*, the cost is directly proportional to the size of node *A*. The entire cost for propagation in a GenSpace graph is directly proportional to the number of records scanned in the propagation process [4]. Instead of reporting the time cost directly, we report the number of the records scanned. This cost is independent of the computer that we use and the details of our implementation.

We conducted experiments to measure the effectiveness of using virtual bottom nodes. We assumed all bottom nodes in ExGen graphs are uninteresting. For the attributes *HighTemp* and *TotalPrep*, which have numeric values and have only one child node of the bottom node in their ExGen graphs, we simply eliminated the bottom node. For attribute *StationID*, we replaced the bottom node *Specific-Station* with a virtual bottom node called *C-D-L-Region*. For attribute *Time*, we replaced the bottom node *Specific-Date* with a virtual bottom node called *YYYYMM-Season*. Table 2 lists the number of the sections in the actual bottom node and the virtual bottom node for the ExGen graphs for the individual attributes and the Genspace graph formed by combining these ExGen graphs. Figure 11 shows percentage of time cost saved by traversing in a GenSpace graph with a virtual bottom node compared with traversing in the original GenSpace graph. We can see that the percentage of the propagation time saved increases as the size of the bottom node increases. Figure 12 compares the time cost and space cost with virtual and real bottom nodes. The X- axis denotes the size of the bottom node, and Y-axis denotes the propagation cost

Table 2. Comparison of the Sizes of the Real Bottom and Virtual Bottom Nodes

Size	Station	Date	HighTemp	TotalPrep	GenSpace
Bottom node	30	18262	152	280	211584
Virtual bottom node	21	1200	4	4	43808

and space cost in thousands of records, respectively. All costs increase linearly with the size of the bottom node, but the rate of increase is smaller for the virtual bottom node than for the real bottom node.

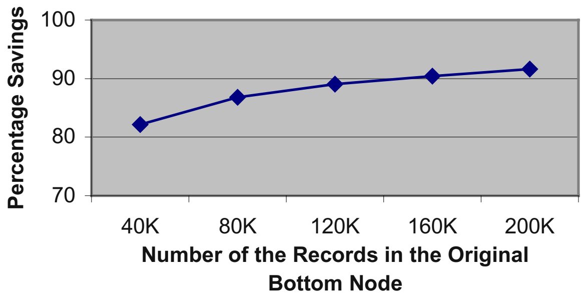
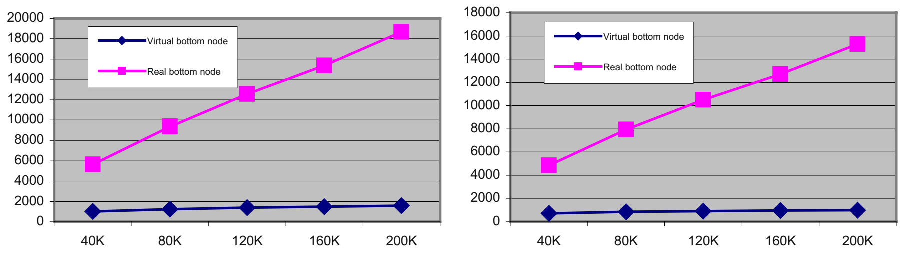
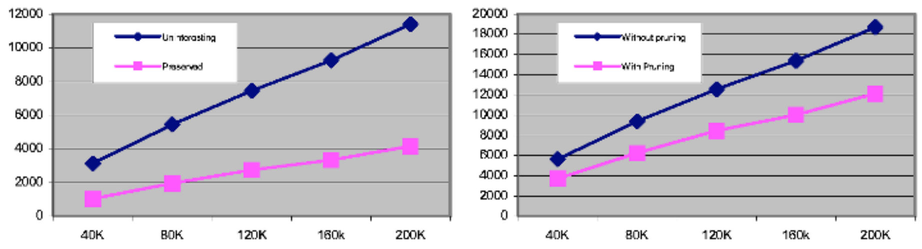


Fig. 11. Percentage Time Savings for Virtual Bottom Nodes.



(a) Propagation Time Cost (b) Space Cost

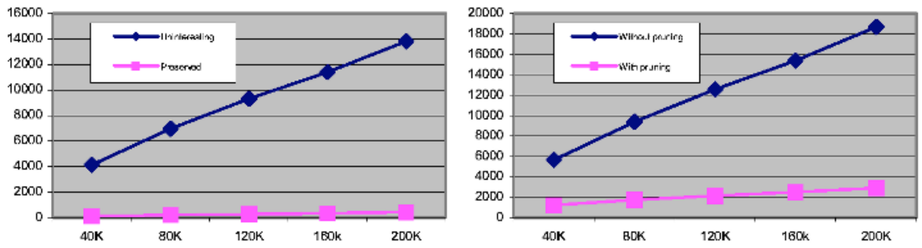
Fig. 12. Virtual Bottom Node vs. Real Bottom Node.



(a) Storage of Uninteresting Nodes versus Storage of Preserved Nodes.

(b) Scanning Cost of Original Graph versus Pruned Graph.

Fig. 13. Scenario 1.



(a) Storage of Uninteresting Nodes versus Storage of Preserved Nodes.

(b) Scanning Cost of Original Graph versus Pruned Graph.

Fig. 14. Scenario 2.

For our experiments on pruning nodes from GenSpace graphs, we assumed two scenarios for the Saskatchewan weather data set. First, we assumed that all the nodes with depth less than or equal to four in GenSpace graph are not interesting. In this case, 165 out of 560 nodes are uninteresting. In the second scenario, we assumed that all nodes with specific date and specific temperature values are not interesting. In this case, 200 nodes are uninteresting. Figure 13(a) shows that the storage space (in thousands of records) for uninteresting nodes and the storage for preserved uninteresting nodes using heuristics for the first scenario, with the size of the bottom nodes ranging from 40K and 200K. Figure 13(b) shows the scanning costs for the GenSpace graph with and without pruning. Figure 14 shows the corresponding trends for the second scenario. In these two scenarios, both storage and scanning costs are significantly reduced when we use our pruning strategy.

7 Conclusions and Future Work

We have tackled several key issues in achieving greater efficiency when traversing a GenSpace graph and finding interesting summaries in it. The efficiency issue is directly related to the number of the nodes in a GenSpace and the number of the records in each node. To reduce the number of the records we have to scan, we proposed three pruning methods for different mining steps. Experiments show that these strategies can improve the search efficiency significantly. At present, we only report simple rules on discrepancies between observations and expectations from interesting summaries to users; in the future, we will look for more expressive means to represent mining results, such as graphs and correlation analysis.

References

[1] Beyer, K. and Ramakrishnan, R., Bottom-up computation of sparse and iceberg CUBEs. *Proceedings of ACM SIGMOD*, 359-370, 1999.

[2] Cai, Y., Cercone, N., and Han J., Attribute-oriented Induction in Relational Databases. In Piatetsky-Shapiro, G. and Frawley, W.J. (eds), *Knowledge Discovery in Databases*, AAAI Press, 1991, 213-228.

[3] Fayyad, U.M., Piatetsky-Shapiro, G., and Smyth, P., From Data Mining to Knowledge Discovery: An Overview. In Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, R., and Uthurusamy, R. (eds), *Advances in Knowledge Discovery and Data Mining*, AAAI Press, 1996, 1-34.
[4] Geng, L and Hamilton, H. J., *Expectation Propagation in ExGen Graphs for Summarization*, Tech. Report CS-2003-03, Department of Computer Science, University of Regina, May 2003.
[5] Hamilton, H.J., Geng, L., Findlater, L, and Randall, D.J., Spatio-Temporal Data Mining with Expected Distribution Domain Generalization Graphs, In *Proceedings 10th Symposium on Temporal Representation and Reasoning / International Conference on Temporal Logic (TIME-ICTL 2003)*, IEEE CS Press, Cairns, Australia, July, 2003, pp. 181-191.
[6] Hamilton, H.J., Hilderman, R.J., and Cercone, N.. Attribute-oriented Induction Using Domain Generalization Graphs. In *Proc. Eighth IEEE International Conference on Tools with Artificial Intelligence (ICTAI'96)*, 246-253, Toulouse, France, November 1996.
[7] Harinarayan, V., Rajaraman, A., and Ullman, J. D., Implementing data cubes efficiently. *Proceedings of ACM SIGMOD* '96, 205-216, 1996.
[8] Sarawagi, S., Explaining Differences in Multidimensional Aggregates. In *Proc. of the 25th Int'l Conference on Very Large Databases* (VLDB), 1999.
[9] Sarawagi, S., Agrawal, R., and Megiddo, N., Discovery Driven Exploration of OLAP Data Cubes. In *Proc. Int. Conf. of Extending Database Technology* (EDBT'98), March 1998.
[10] Randall, D.J., Hamilton, H.J., and Hilderman, R.J., Temporal Generalization with Domain Generalization Graphs, *International Journal of Pattern Recognition and Artificial Intelligence*. 13(2):195-217, 1999.
[11] Yao, Y.Y. and Zhong, N., Potential applications of granular computing in knowledge discovery and data mining, *Proceedings of World Multiconference on Systemics, Cybernetics and Informatics, Volume 5, Computer Science and Engineering*, Orlando, Florida, USA, 573-580, 1999.
[12] Zytkow, J., From Contingency Tables to Various Forms of Knowledge in Databases. In Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, R., and Uthurusamy, R., *Advances in Knowledge Discovery and Data Mining*, AAAI Press, 329-349, 1996.

Appendix: Proof for Theorem 1

Theorem 1
In a GenSpace graph, a node P consists of partition $\{S_1, S_2, \ldots, S_n\}$, and we use the relative variance as the interestingness measure. If for every partition S_i we have $d(S_i) = \frac{|e(S_i) - o(S_i)|}{o(S_i)} \le t,\ t \ge 0$, the relative variance of all descendents of P is less than t.

Proof.
Because $\frac{|e(S_i) - o(S_i)|}{o(S_i)} \le t$, for any i, we have relative variance for node P

$$Intr\ (P) = \frac{1}{n}\sum_{i=1}^{n}\frac{|e(S_i) - o(S_i)|}{o(S_i)} \le t \cdot$$

Suppose node Q is a descendent of node P. We have the partition for $Q = \{T_1, T_2, \ldots, T_m\}$, where $T_i = \bigcup_{S_j \subseteq T_i} S_j$,

$$\frac{|e(T_i) - o(T_i)|}{o(T_i)} = \frac{|\sum_{S_j \subseteq T_i} e(S_j) - \sum_{S_j \subseteq T_i} o(S_j)|}{\sum_{S_j \subseteq T_i} o(S_j)} = \frac{|\sum_{S_j \subseteq T_i} (e(S_j) - o(S_j))|}{\sum_{S_j \subseteq T_i} o(S_j)} \le \frac{\sum_{S_j \subseteq T_i} |e(S_j) - o(S_j)|}{\sum_{S_j \subseteq T_i} o(S_j)}$$

Because for any nonnegative real value x_i and y_i, we have $\frac{\sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} y_i} \le \max(\frac{x_i}{y_i})$,

We get $\frac{|e(T_i) - o(T_i)|}{o(T_i)} \le \max_{S_j \subseteq T_i} (\frac{|e(S_j) - o(S_j)|}{o(S_j)}) \le t$,

Therefore, $Intr(Q) = \frac{1}{m} \sum_{i=1}^{n} \frac{|e(T_i) - o(T_i)|}{o(T_i)} \le t$.

Naïve Bayes with Higher Order Attributes

Bernard Rosell[1] and Lisa Hellerstein[2*]

[1] Polytechnic University, Dept of Computer and Information Science,
6 Metrotech Center, Brooklyn, NY 11201 USA,
brosell@att.com

[2] Polytechnic University, Dept of Computer and Information Science,
6 Metrotech Center, Brooklyn, NY 11201 USA,
hstein@cis.poly.edu

Abstract. The popular Naïve Bayes (NB) algorithm is simple and fast. We present a new learning algorithm, Extended Bayes (EB), which is based on Naïve Bayes. EB is still relatively simple, and achieves equivalent or higher accuracy than NB on a wide variety of the UC-Irvine datasets. EB is based on two ideas, which interact. The first is to find sets of seemingly dependent attributes and to add them as new attributes. The second idea is to exploit "zeroes", that is, the negative evidence provided by attribute values that do not occur at all in particular classes in the training data. Zeroes are handled in Naïve Bayes by smoothing. In contrast, EB uses them as evidence that a potential class labeling may be wrong.

1 Introduction

Naïve Bayes (NB) is one of the most popular learning algorithms. It is simple and fast, achieves reasonably good accuracy, and is based on a clean probabilistic model. However, more complex learning models can often achieve greater accuracy. This is due in part to the limited expressiveness of the hypothesis produced by NB [8]. A related problem is that NB is based on independence assumptions that are often violated in real-world problems. Although on many datasets NB achieves good accuracy in practice despite these violations [7], on other datasets increased accuracy can be obtained by taking dependencies into account.

There have been a variety of attempts to modify NB to improve its accuracy. Some of the resulting algorithms are general-purpose, while others are tailored to particular applications such as text categorization (cf. [10,13,16,25,31]). In this paper we present a new, modified version of NB, which we call extended Bayes (EB). The EB algorithm is general-purpose, and it achieves as good or better performance than the standard version of NB on a wide variety of UC-Irvine datasets [4]. Compared to many previous modifications of NB, EB is conceptually simpler. In contrast to NB

* Partially supported by NSF Grant CCR-9877122.

itself, and to some modified versions of it, EB is heuristic and does not correspond to a probabilistic model.

In the remainder of this section we provide an overview of related work, a brief description of the standard NB algorithm, and some motivation for our approach. In Section 2 we describe the EB algorithm. Finally in Sections 3 and 4 we present our experimental results.

1.1 Related Work

The Machine Learning literature contains many modifications to NB. In this section, we describe some of the modified versions of NB that are most relevant to our algorithm, EB.

A standard approach to improving NB is to find and exploit dependencies between variables. Kononenko's Semi-Naïve Bayesian classifier [16] searches all pairs of attribute values and if the values are shown to be statistically dependent, they are joined together to form a new feature. Experimental results using this classifier were not decisive. Pazzani [24] used cross validation techniques to find the attribute pair that, when joined together into a new extended attribute, yielded the largest increase in accuracy. He iterated this process until no further improvements were obtained. This approach produced positive experimental results. In a similar vein, Sahami [31] defined a extension of NB that allowed each attribute to depend on at most a predefined number of other attributes. The TAN algorithm of Friedman et al. [10] and its variants [13,18] build a restricted Bayesian network classifier that encodes dependencies between attributes in a tree structure and can be learned in polynomial time.

Another approach to modifying NB has been to exploit techniques from data mining. Both the CBA algorithm of Liu et. al. [19], and the Large Bayes algorithm of Meretakis et. al. [21,22] are based on finding informative patterns in the training data. The CBA algorithm works by finding class association rules (CARs). Essentially, a CAR is a mapping between an itemset (combination of attribute values) and a class label. A new instance is classified by applying the best rule that matches the instance. The Large Bayes algorithm works by finding itemsets that are both frequent (reliable) and interesting (evidential). A test example **y** is classified by combining the evidence from each of the itemsets found in the training phase that match **y**. This evidence is simply the conditional probability of the class given the itemset. Both CBA and Large Bayes have shown positive experimental results

The EB algorithm is similar to CBA and Large Bayes in that it uses data-mining techniques to perform a heuristic search for useful higher-order attributes. However, there are many differences. Note that Large Bayes and CBA search for itemsets, which are combinations of attributes with associated values, rather than for combinations of attributes. Also, CBA and Large Bayes select a distinct collection of itemsets for each class, whereas EB selects one set of attributes (both original and higher-order) to be used for all of the classes. In Large Bayes and CBA, the support (frequency) of an itemset within a class is a crucial factor used in determining whether to select a particular itemset. In EB, information gain, a discriminative measure, is used to determine whether a collection of attributes should be selected. In practice, Large Bayes finds and uses very large itemsets (hence the name Large Bayes), as does CBA. The search for long itemsets makes Large Bayes and CBA computationally intensive. Theoretically, EB can find and use large collections of attributes. However, in the experiments we performed on a wide selection of UCI datasets, the stopping criteria for the heuristic search in EB caused it to stop before examining any collection of more than four attributes. Finally, we believe that EB is conceptually simpler than both CBA and Large Bayes.

Overall, EB lies somewhere between the more standard Machine Learning algorithms mentioned above, which concentrate on dependent attribute sets, and the data-mining based algorithms, which concentrate on per-class itemsets. A novel aspect of EB, not shared by any of the above algorithms, is its exploitation of the "zeroes". This is discussed more fully below.

We designed EB to be a general-purpose version of NB and tested it on UCI datasets. A question for future research is whether a version of EB, adapted for text, would perform well on text learning problems; versions of NB are already widely used for problems such as text categorization and spam detection (cf. [14, 17, 25, 29, 30, 32]).

1.2 Overview of NB

We begin by providing a brief overview of the standard NB algorithm. A more detailed description can be found in [23].

The NB algorithm is based on the following probabilistic model. Let $\{C_1, ..., C_m\}$ be a set of m classes. Consider a (multi) set X of labeled examples, each consisting of an n-dimensional attribute (feature) vector $<y_1, \ldots, y_n>$ assigning values to the attributes $A_1,\ldots,A_n$. Let D be a probability distribution over X. Then, given a random sample X' drawn from X and a new unlabeled example y = $<y_1, \ldots ,y_n>$ drawn from the distribution D, the maximum likelihood prediction for y is:

$$\underset{C_i}{Arg\max} \quad P_D(C_i \mid y_1,\cdots,y_n), \tag{1}$$

where P_D is the conditional probability measure on $\{C_1, ..., C_m\}$ induced by D. The NB assumption is that the attributes are independent when conditioned on the class label. Using this assumption, an equivalent formula for the maximum likelihood prediction can be shown to be

$$\underset{C_i}{Arg\max} \quad P(C_i) * \prod_{j=1}^{n} P(y_j \mid C_i)\,. \tag{2}$$

Given an unlabeled example and set of training data, NB uses this maximum likelihood estimate (MLE) to predict the label of the example. More particularly, it estimates the probabilities in the above expression using the training data and determines the maximum likelihood class based on these estimates. In what follows, we use $P(C_i)$ and $P(y_j| C_i)$ to denote empirical probabilities, which are equal to the frequencies observed in the training data.

A technicality of the NB implementation is how probabilities are estimated given the training data. Although it is possible to use the empirical probability $P(y_j| C_i)$ in place of $P_D(y_j| C_i)$ in calculating the above expression, it is problematic to do so when $P(y_j| C_i)$ is equal to zero. As a result, "smoothing" or other error correction techniques are often used to modify empirical frequencies and avoid zero values.

There are several methods for avoiding zero values [15]. The benchmark version of NB discussed in this paper uses the so-called No-Match method [6,15]. This method assumes that with larger samples the zero terms would converge to some small non-zero value. If $P(y_j \mid C_i) = 0$, the estimate $P(C_i) / N$, where N is the sample size, is used instead. In what follows, whenever we refer to "zeroes", we mean empirical probabilities $P(y_j \mid C_i)$ that are equal to 0.

1.3 Additional Notation and Terminology

Given a set *A* of attributes $\{A_j\}$, define an higher order attribute of order k to be a subset of $\{A_j\}$ of size k. Let $y = <y_1, .., y_n>$ be an unlabeled example and let $\mu = <A_{j1}, A_{j2}, \dots, A_{jk}>$ be a higher order attribute of order k. Then we denote the value of μ on the instance y by $\mu(y) = <y_{j1}, y_{j2}, \dots, y_{jk}>$. For any set Q of higher order attributes, the expression $\{\mu(y) \mid \mu \varepsilon Q\}$ denotes the set of itemsets for this example. The MLE expression in eq.2 can be calculated over the set Q instead of the initial set of attributes, yielding the following expression:

$$\underset{C_i}{Arg\max} \quad P(C_i) * \prod_{\mu \in Q} P(\mu(y) \mid C_i) . \tag{3}$$

EB sometime bases its calculations on just such an expression.

1.4 Motivation

In problems where there is dependence between the variables, NB may produce the wrong prediction. Consider the following variant of an example from Rachlin et al. [28]. Let M = {0.0, 0.01, …, 0.99, 1.0} and note that M^2 is a discrete subset of the unit square. There is a linear separator defined on M^2 represented by the equation y = ax ($a \neq 1$). Points in M^2 are in class *A* if they are above the line y = ax and in class B if they are below. Suppose that all the points in M^2 are in the training set for the resulting classification problem, ensuring that $P(y_j|C_i) = P_D(y_j |C_i)$, assuming D is the uniform distribution.

Rachlin et al. showed that for a = 3, NB would incorrectly classify precisely those points in the narrow area bounded by the functions y = 3x and y = 9x/(5 –12x). For example, the point <.90, .25> would be incorrectly classified by NB as belonging to class B. For points in this area, both the y and x coordinates individually are poor predictors of class membership. In contrast, the joint values of the x and y coordinates are perfect predictors of class membership. In addition, simply knowing that $P(<.9, .25> | B) = 0$ disqualifies class B, leaving A as the only remaining choice.

This simple example suggests two main approaches to extending NB, which are the basis for the EB algorithm. First, modify NB to use combinations of the initial attributes. Some of these combinations may be highly informative. Second, modify NB to take zeroes seriously. More particularly, if some attribute values never occur in conjunction with class C_i in any of the training examples, and there is sufficient training data, then the presence of those attribute values in a test example is evidence that C_i is not the correct prediction. The evidence is particularly strong if those same attribute values occur frequently in training examples associated with other classes. If the evidence is strong enough, the prediction of C_i should be disqualified.

2 Overview of EB

In the training phase, EB begins by finding sets of attributes that appear to have high predictive power. The search for these sets is based on data-mining techniques, and is described in detail below. EB takes each set that is found and defines a new higher order attribute made up of the attributes in the set. Each new attribute is added to the original attribute set, yielding an extended attribute set. EB then calculates the estimates for $P(C_i)$ and $P(y_j|C_i)$, for each class C_i, and each relevant value y_j of each attribute A_j in the extended attribute set.

Given a test example, EB generates two possible predictions for the class label of the example, an initial prediction, and an alternative. The initial prediction is just the one that would be calculated by using standard NB on the original attribute set as in eq. 2 The alternative prediction is also calculated as in standard NB, except that calculations are performed using the extended attribute set, as in eq. 3, rather than using the original attribute set.

Finally, EB decides whether to use the initial NB prediction or the alternative prediction. To make its decision, it checks whether either of the following two conditions is satisfied:

1. There exist sets of (original or higher-order) attributes whose values in the test example have never been observed in any training example from the class predicted by NB. These "zeroes", taken together, constitute strong evidence that the initial NB prediction is incorrect.
2. There exists at least one higher-order attribute whose information gain is very large.

The first condition essentially disqualifies the initial NB prediction. The second indicates strong dependence between attributes, and thus supports the use of the alternative prediction. If either of the two conditions is satisfied, EB uses the alternative prediction. Otherwise, it uses the initial prediction.

Intuitively, EB exploits dependence between attributes (given the class label) when it exists, and avoids making mistakes when dependence does not exist. When dependence exists, higher-order attributes are informative both in indicating the correct class and in disqualifying incorrect classes. When dependence between attributes doesn't exist, higher-order attributes may capture only spurious correlations. In this case, the checks made by EB should cause it to use the standard NB prediction, based only on the original attributes, rather than the alternative prediction.

2.1 Generating the Higher Order Attributes

The choice of which higher-order attributes should be placed in the extended attribute set is a delicate one. When there are too many higher-order attributes, they tend to produce many zeroes, causing EB to disqualify the initial NB prediction (through Condition 1 above), whether it is right or wrong. The higher the order of the attribute, the more likely it is, statistically, to produce a zero. On the other hand, our experiments indicate that zeroes can be very helpful in determining when to disqualify the initial NB prediction. If few or no higher-order attributes are generated, EB will not do better than NB.

The search procedure used by EB to find higher-order attributes proceeds in steps or levels. Let A be the original attribute set. The extended attribute set EXT(A) is initialized to be equal to A. In each level k, the procedure calculates empirical conditional probabilities, using the training set, for the attributes of order k (i.e. made up of k attributes) in EXT(*A*). It then determines the attributes of order k+1, and proceeds to level k+1.

The goal of the search is to generate a manageable set of extended attributes that will help to distinguish between the various classes. The techniques employed to guide this search are modeled on those used in various data mining and inductive classification algorithms [1,2,3,5,26,27].

EB uses the information gain measure to determine, at each level, which set of candidate attributes to use to form higher order attributes. Let H(C) denote the class entropy and H(C| A) denote the conditional entropy of C given attribute A. The information gain I(C|A) is defined to be H(C) - H(C| A) (see [23] for more details). Information gain proves useful because it measures the ability of an attribute to distinguish between classes. The percentage reduction in the uncertainty of the class label provided by an attribute A is given by I(C|A) / H(C).

To constrain the inherent combinatorial explosion, EB implements the following standard monotonic pruning procedure. In each level k, EB selects as "candidates" those attributes in EXT(A) that yield a reasonable value, denoted $\zeta(k)$, for the percentage reduction in uncertainty of the class label. By using attributes of increasingly higher order, one hopes to get higher predictive power and therefore $\zeta(k)$ is implemented as an non-decreasing function of k. Given two candidate attributes of order k, let U equal the union of their component elements. Each subset B of U of size k+1 is an attribute of order k+1. EB places each such attribute B in EXT(*A*) iff every subset of B of size k is a candidate for expansion.

Some higher order attributes that are placed in EXT(*A*) in this process may yield a large reduction the uncertainty of the class label. If a variable B yields a reduction in uncertainty greater than a pre-defined function $\phi(k)$, which is also non-decreasing in k, we call B a strongly predictive attribute. Finding such attributes when they occur is a measure of success for this approach.

Finally, to avoid stopping immediately at level 1, EB always ensures that there are at least a small number of attributes in the level 1 candidate set.

2.2 Evaluating Events with Zero Probability

Let $y = <y_1, ..,y_n>$ be an unlabeled example and let $\mu = <A_{j1}, A_{j2}, \dots , A_{jk}>$ be a higher order-attribute in EXT(*A*).

Suppose that for some class C_i, unlabeled example y, and some μ ε EXT(*A*), the empirical probability of $P(\mu(y) \mid C_i)$, calculated from the training set, is 0. This provides some evidence that the attribute value $\mu(y)$ never occurs in class C_i, and hence C_i cannot be a correct prediction for example y. However, the evidence may not be strong. To assess the strength of the evidence, consider a hypothesis test using two candidate hypotheses. Take as a null hypothesis, H_0, that the true probability of $\mu(y)$ occurring in class C_i is $P(\mu(y))$. That is to say, there is nothing special about itemset $\mu(y)$ with respect to class C_i – it occurs with the same frequency in examples from C_i as it does in the general population (and we use the empirical probability $P(\mu(y))$ as our estimate of the frequency of $\mu(y)$ in the general population). The alternative hypothesis H_1 is that the true probability of $\mu(y)$ is 0, that is, $\mu(y)$ never appears in any examples in class C_i. The training set provides the data for the hypothesis test. We reject H_0 in favor of H_1 iff $\mu(y)$ does not appear in any training examples belonging to class C_i.

If m is the number of examples from class C_i in the training set, then the probability of rejecting hypothesis H_0 when it is actually true is $(1 – P(\mu(y))^{m}$. This is the probability that $\mu(y)$ doesn't appear in any training examples from class Ci when its rate of occurrence in C_i is $P(\mu(y))$. The smaller this probability, the more plausible it is that, if $P(\mu(y) \mid C_j) = 0$, then the zero is "real", that is, $\mu(y)$ never occurs in class C_i.

Therefore, for a class C_i and itemset $\mu(y)$, we define

$$\text{cred}(C_i, \mu(y)) = \begin{Bmatrix} 1 \leftrightarrow P(\mu(y) \mid C_i) \neq 0 \\ (1 - P(\mu(y)))^m \leftrightarrow P(\mu(y) \mid C_i) = 0 \end{Bmatrix}. \tag{4}$$

The smaller the value of $\text{cred}(C_i, \mu(y))$, the stronger our belief that $\mu(y)$ never occurs in class C_i, and that C_i is therefore not a correct prediction for the label of y.

Since a test example y may generate multiple itemsets with zero counts for a given class, we also make the following definition.

$$\text{Eval}(C_i) = \prod_{\mu \in EXT(A)} \text{cred}(C_i, \mu(y)) . \tag{5}$$

Eval(C_i) aggregates the information from all the zeroes associated with itemsets generated from y, so that the cumulative evidence from many itemsets can be evaluated.

EB uses Eval(C_i) in deciding whether to reject the initial prediction C_i produced by running standard NB on a test example. If Eval(C_i) < P_T for some chosen threshold value P_T, then EB rejects C_i and predicts using the alternative prediction generated from the extended attribute set. Clearly, the value of P_T directly controls the algorithm's performance. In doing preliminary experiments, we did not find the performance of EB to be particularly sensitive to the choice of P_T. Our final experiments used a fixed value for P_T. To make the algorithm more robust, it might be preferable to choose the value of P_T in a separate tuning phase, but we achieved good results across many datasets with a fixed value.

2.3 The EB Algorithm

An outline of the algorithm is presented below. The training phase is described first.

1. Initially, k = 1 and EXT(A) = {{A_j}}, the set of initial attributes. Let Candidate-Set = {A_j}. For each class C_i, compute the empirical probability P(C_i) from the training data.
2. For each attribute A_j in Candidate-Set and each class C_i, compute estimates for P(A_j) and P(A_j | C_i) by computing empirical frequencies based on the training data.
3. Check each attribute A_j of order k in Candidate-Set to determine whether I(C|A) / H(C) < ζ(k), and if so, remove A_j from Candidate-Set. If k =1 and Candidate–Set = ∅, then keep a small percentage of attributes with the largest information gain to jump-start the process.
4. For each pair A_i and A_j of attributes remaining in Candidate-Set, find all subsets B of size k+1 contained in the union of A_i and A_j. Add each such B to EXT(A) as a new attribute of order k+1 iff every subset of B of size k is in Candidate-Set.
5. Update Candidate-Set to include only the new attributes of order k+1 just added to EXT(A).
6. Check each attribute A_j of order k +1 in Candidate-Set to determine whether I(C|A) / H(C) > ϕ(k). If so, mark it as strongly predictive.
7. If Candidate-Set is not empty, increment k and return to step 2. Otherwise, proceed to the test phase.

Fig. 1. Training Phase

The test phase uses the information generated in the training phase.

1. Given an example y, compute the maximum likelihood class C_i (eq. 2) over the initial set of attributes. C_i is the initial prediction. Calculate Eval(C_i).
2. For the same example y, compute the maximum likelihood class C_k (eq. 3) over the extended set of attributes. Class C_k is the alternative prediction.
3. Disqualify the initial prediction C_i if there exists a strongly predictive extended attribute that predicts a different class or if Eval(C_i) < P_T.
4. If the initial prediction is disqualified, output the alternative prediction Otherwise, output the initial prediction.

Fig. 2. Test Phase

3 Experimental Section

3.1 Parameter Settings

In our experiments on the UCI datasets we made sure at level one of the attribute generation process that there were always a few elements in the candidate set but not too many (step 3 of the training phase). Without enough candidates the algorithm would default to NB and with too many candidates the run time could potentially blow up. We take the t attributes with the largest information gain where t = min(lg |initial attributes| + 1, 16).
$\zeta(k)$ and $\phi(k)$ were set as follows:

Table 1. Parameter setting of EB and SEB

K	$\zeta(k)$	$\phi(k)$
1	0.10	NA
2	0.33	0.88
3	0.50	0.95
> 3	0.75	0.98

3.2 Results

NB and EB were tested on a variety of classification problems obtained from the UCI repository [4]. In choosing the datasets we attempted to vary the number and type of attributes, dataset size, number of classes, etc. The resulting 16 data sets are among the most commonly studied in the machine learning community.

Continuous attributes were discretized by partitioning their range into 15 uniform intervals. A few of the training sets had instances where there were missing values for some attributes. In these cases a special character ("?") was inserted into the training and test examples. Both algorithms handle these cases in a fairly primitive manner, i.e., they treat the special character as an additional attribute value.

A 10 fold cross validation testing mode was used for all datasets with the exception of the Letter, DNA, Soybean and Monk datasets. For these datasets the UCI repository provides pre-defined train and test sets.

In all test runs the order of the attributes never exceeded 4, a relatively small value. Indeed, significant information can be obtained by considering attributes of order 2 or order 3. One likely reason for this is that the hypothesis to be discovered can in many cases be very closely approximated by a small set of simple rules, each involving only a few attributes [11].

We also tested a variant of EB that we call SEB, Simple Extended Bayes. SEB always predicts using the alternative hypothesis, which, as described above, is the result of applying eq. 3 to the extended attribute set. Since SEB does not choose between the initial NB hypothesis and the alternative, it makes no use of information from zeroes or from the strongly predictive attributes.

Table 2 gives the results of these experiments. The average accuracy (percentage of test examples that were correctly classified) and standard deviations (where appropriate) are provided for all three algorithms.

Table 2. Classification accuracy for Naïve Bayes, Extended Bayes and Simple Naïve Bayes on test data sets

Data Set	NB	EB	SEB
House	90.0 ± 4.2	93.6 ± 3.3	94.4 ± 2.9
Monk-1	74.6 ± 4.4	91.6 ± 13.7	91.6 ± 13.7
Monk-2	62.4 ± 3.2	62.4 ± 3.2	50.8 ± 5.8
Monk-3	96.2 ± 2.2	98.8 ± 1.4	98.8 ± 1.4
Car Evaluation	86.2 ± 2.1	89.0 ± 2.3	90.2 ± 2.1
DNA	93.3	93.8	89.1
Mushroom	99.2 ± 0.03	99.9 ± 0.01	99.8± 0.02
Adult	99.5	99.8	99.7
Vehicle	60.4 ± 5.2	70.1 ± 3.8	70.3 ± 4.5
Soybean	92.50	94.1	94.1
German	74.9 ± 3.8	75.4 ± 3.6	72.4 ± 3.6
Promoter	88.8± 9.1	94.5 ± 9.5	94.5 ± 9.5
Ecoli	81.8 ± 7.3	82.0 ± 9.0	80.0 ± 8.1
Tic-Tac-Toe	69.6 ± 3.5	75.9 ± 3.0	67.8 ± 4.2
Connect4	72.5± 0.3	72.6± 0.3	56.5± 1.3
Nursery	90.2 ± 0.7	90.4 ± 0.8	86.3 ± 1.2
Letter	74.9	90.7	91.9
iris	93.3 ± 5.9	96.8 ± 5.0	96.8 ± 6.0
Pima	75.7 ± 4.0	75.7 ± 4.0	73.1± 6.1
Heart	82.0 ± 6.1	81.9 ± 6.9	75.6 ± 7.8

For the Monk-2, Pima Diabetes, Heart, Ecoli and German Credit data sets, EB and NB have essentially identical accuracy. For the other data sets, EB achieves higher accuracy. The degree of improvement runs the gamut from minimal to very large. It is interesting that SEB does better than EB on those datasets where EB has a clear edge, no matter how small, over NB. However, on the other datasets where NB and EB perform equivalently, SEB performs significantly worse than either NB or EB. On such datasets, EB can weigh the strength of the zeroes and default to the NB prediction if the zeroes are weak. SEB is forced to predict the alternative prediction. The fact that EB consistently does as well or better than NB, and doesn't do worse than SEB, suggests that the criteria used by EB to decide between NB prediction and the alternative prediction are effective.

The basic reason is that SEB performs well in the presence of zeroes, i.e., the net effect of all the tiny zero corrections ends up disqualifying the NB prediction based on only the original attribute set. However, when there are no zeroes or only very weak ones and the training phase did not uncover highly predictive extended attributes, the extra information from the extended attribute set is mostly noise.

We discussed the idea of strongly predictive extended attributes and indicated that it would be highly desirable for the training phase to find them if they exist. In the Monk1 and Monk3 datasets, such attributes of order 3 exist by definition. The approach we have outlined here found them without having to examine all attributes of order 2 or order 3. Strongly predictive attributes were also found in the Iris and Mushroom datasets. It is important to note that such situations are not commonplace and the improvements obtained by EB in most cases do not rely on finding such attributes but instead rely on evidence provided by the zero counts.

By itself, classification accuracy can be misleading for a variety of reasons when comparing learning algorithms [9, 18]. To address some of the shortcomings other authors have provided variable cost metrics or ROC analyses. These measures are difficult to apply when the problem involves more than two classes. Given that we are only comparing two related algorithms we take the following approach.

In Table 3 we provide the following information for each dataset we used:

1. The total number of test examples on which each classifier was tested. (If cross validation was used, this equals the numbers of examples in the dataset.)
2. The number of times that EB made a different prediction than NB and NB was actually correct.
3. The number of times that EB made a different prediction than NB and EB was actually correct.

In some cases where the absolute accuracy improvement (Table 1) is minimal, as in the mushroom dataset, the relative improvement is significant. In the mushroom trial EB correctly classified more mushrooms on each trial and **always** (Table 2) gave a better answer than NB. (If you eat a lot of mushrooms this could potentially have life-saving implications.) For all the datasets this improvement is consistent across each of the trials, i.e., EB performs as well or better than NB on almost every experiment. The improvement for the Letter, Vehicle, and Monk1 datasets significantly improves upon the performance of NB.

Table 3. Performance for NB and EB on individual test examples

DataSet	# Test Examples	# (NB > EB)	# (EB > NB)
House	435	1	15
Monk-1	556	0	93
Monk-2	601	0	0
Monk-3	554	0	14
Car Evaluation	1728	0	48
DNA	1186	1	7
Mushroom	8,124	0	58
Adult	16,281	0	36
Vehicle	846	16	97
Soybean	367	1	7
German	1,000	3	9
Promoter	105	2	8
Ecoli	336	0	2
Tic-Tac-Toe	959	6	67
Connect 4	67,558	6	64
Nursery	12,959	0	18
Letter	5,000	31	822
Iris	150	0	3
Pima	768	0	0
Heart	291	2	1

In Table 4 we present the results of some preliminary experiments comparing EB to three different types of classification algorithms: support vector machines (SMO), Bayes nets (BayesNetB) and a class association rule mining algorithm (CBA). We used the WEKA toolkit [32] for SMO and BayesNetB. For CBA (discussed in Section 1.1) we use the software package CBA 2.0 of Liu et. al. [19]. For SMO and BayesNetB we used the default parameter settings in the WEKA package. For CBA, we experimented both with the recommended parameter settings and some alternatives. In Table 4 we report the results with the default/recommended parameter settings except in the case of the Nursery and Letter datasets. (The results we present for CBA on the Letter and Nursery datasets were achieved with only single class minimum support. For the Letter dataset, we were not able to run CBA with the recommended setting and for the Nursery dataset we achieved much higher accuracy with the alternative setting.)

For several of the larger datasets, such as Letter, Connect4 and Adult, we were not able to generate the learning models for the SMO and BayesNetB algorithms in WEKA or for CBA due to memory limitations.

On most datasets, the four algorithms have similar accuracy. SMO had much higher accuracy on the Tic-Tac-Toe dataset. For the Car and Nursery datasets, both SMO and BayesNetB had much higher accuracy than the other two algorithms.

Table 4. Comparison of EB with other classification algorithms

Data Set	EB	SMO	BayesNetB	CBA
House	94.1	96.0	94.1	95.0
Monk-1	91.6	72.2	98.6	96.8
Monk-2	62.4	65.7	63.0	74.6
Monk-3	99.0	97.6	96.3	98.6
Car Evaluation	89.6	93.6	93.5	87.0
DNA	93.8	93.0	-------	90.0
Mushroom	99.9	100.0	100.0	98.1
Adult	99.8	------	-------	99.8
Vehicle	70.7	71.3	71.9	65.8
Soybean	94.1	91.8	95.4	87.2
German	75.4	76.7	74.6	70.0
Promoter	93.6	93.4	81.8	72.4
Ecoli	80.9	81.2	80.7	62.3
Tic-Tac-Toe	76.0	98.3	79.7	85.8
Connect4	72.6	-------	--------	--------
Nursery	90.4	93.0	94.18	82.9
Letter	90.7	-------	--------	65.0
iris	96.9	95.3	96.6	93.3
Pima	76.7	75.0	77.3	65.1
Heart	79.0	81.1	82.0	80.8

Carefully choosing the parameter settings for the above algorithms rather than using the default settings can yield better accuracy results. For example, support vector machines depend on many parameters including the kernel function, kernel parameter and cost parameter. Hsu and Lin [32] used cross validation to determine parameter settings for multi-class support vector machines. (They preprocessed the data in a different way than we did). On several datasets they reported much higher accuracy than we obtained in our SMO and non-SMO experiments. Most notably, they reported accuracies of 97.9% and 86.6% on the Letter and Vehicle datasets respectively. We have not yet experimented with parameter tuning for EB; presumably EB would achieve higher accuracy through parameter tuning, although we suspect it would not achieve the accuracies reported by Hsu and Lin.

4 Conclusions

EB is a promising alternative to NB when the higher order attributes contain useful information. It has been noted that relative to other algorithms, NB generally performs poorly on large datasets since additional data points cannot decrease the bias component of the error term [15]. Since EB can benefit from information about higher order attributes, EB has more potential than NB to improve, i.e., achieve smaller bias, as the training set grows larger, without a corresponding increase in the variance of the error term. As the size of the training set increases, so too does the quality of the evidence of the zero counts that are found. In addition, the probability of incorrectly disqualifying the correct class decreases.

Previous evaluations of NB have shown that it compares favorably with the majority of other classification algorithms [7]. The improvements described in this paper provide further evidence of the flexibility and utility of the NB algorithm, and the utility of basic data mining techniques in classification algorithms. EB also demonstrates the benefit of directly exploiting the information from zeroes, rather than just smoothing or correcting them.

References

1. Agrawal, R., Ghosh, S., Imielinski, T., Iyer, B., and Swami, A. An interval classifier for database mining applications. In *Proc. of the 18th VLDB Conference,* pp. 560-573, 1992.
2. Agrawal, R., Imielinski, T., and Swami, A. Database mining: a performance perspective. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):914-925, 1993.
3. Agrawal, R., Imielinski, T., and Swami, A. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD Conference,* pp. 207-216, 1993.
4. Blake, C. L. and Merz, C. J. UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, 1998.
5. Brin, S., Motwani, R., and Silverstein, C. Beyond market baskets: generalizing association rules to correlations. In *Proc. of the ACM SIGMOD Conference, pp. 265-276,* 1997.
6. Clark, P. and Niblett, T. The CN2 induction algorithm. *Machine Learning*, 3:261-283, 1989.
7. Domingos, P., and Pazzani, M. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29:103-130, 1997.
8. Duda, R. O. and Hart, P. E. *Pattern Classification and Scene Analysis*. New York: Wiley Interscience, 1973.
9. Foster, T., Kohavi, R., and Provost, F. The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the 15th International Conference on Machine Learning* , pp. 445-453, 1998.
10. Friedman, N., Geiger, D. and Goldszmidt, M. Bayesian network classifiers. *Machine Learning*, 29:131-163, 1997.
11. Holte, R. C. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63-91, 1993.
12. Hsu, C. and Lin, C. A comparison of methods for multi-class Support Vector Machines. *IEEE Transactions On Neural Networks,* 13(2):415-425, 2002.
13. Keogh, E. J. and Pazzani, M. J. Learning augmented Bayesian classifiers: a comparison of distribution-based and classification-based approaches. In *Proceedings of the 7th International Workshop on Artificial Intelligence and Statistics,* pp. 225-230, 1999.

14. Keim, M., Lewis, D. D., and Madigan, D. Bayesian information retrieval: preliminary evaluation. In *Preliminary Papers of the 6th International Workshop on Artificial Intelligence and Statistics*, pp. 303-310, 1997.
15. Kohavi, R., Becker, B., and Sommerfield, D. Improving simple Bayes. In *Proceedings of the 9th European Conference on Machine Learning, pp. 78-87,* 1997.
16. Kononenko, I. Semi-naïve Bayesian classifier. In *Proceedings of the 6th European Working Session on Learning*, pp. 206-219, 1991.
17. Lewis, D. D. Naïve (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of the European Conference on Machine Learning*, pp. 4-15, 1998.
18. Ling, C. X. and Zhang, H. Toward Bayesian classifiers with accurate probabilities. In *Proceedings of the 6th Pacific Asia Conference on Knowledge Discovery and Data Mining*, pp. 123-134, 2002.
19. Liu, B., Hsu, W., & Ma, Y. Integrating classification and association rule mining. *Proceedings of the 4th ACM SIGKDD Conference, pp. 80-86,* 1998. [http://www.comp.nus.edu.sg/~dm2/result.html].
20. McCallum, A. and Nigam, K. A comparison of event models for Naïve Bayes classification. In *Proceedings of the AAAI Workshop on Learning for Text Categorization,* 1998.
21. Meretakis, D. and Wuthrich, B. Extending Naïve Bayes classifiers using long itemsets. *Proceedings of the 5th ACM SIGKDD Conference*, pp. 165-174, 1999.
22. Meretakis, D., Hongjun, L. & Wuthrich, B. A Study on the performance of Large Bayes Classifier. In *Proceedings of the 11th European Conference on Machine Learning*, pp. 271-279, 2000.
23. Mitchell, T. *Machine Learning.* San Francisco: McGraw-Hill, 1997.
24. Pazzani, M. Searching for dependencies in Bayesian classifiers. *Artificial Intelligence and Statistics IV, Lecture Notes In Statistics.* New York: Springer Verlag, 1995.
25. Peng, F. and Schuurmans, D. Combining Naïve Bayes and n-gram Language Models for Test Classification. In *F. Sebastiani (Eds): Advances in Information Retrieval: Proceedings of the 25th European Conference On Information Retrieval Research,* pp. 335-350, 2003.
26. Quinlan, J. R. Induction of decision trees. *Machine Learning,* 1:81-106, 1986.
27. Quinlan, J. R. Simplifying decision trees. *International Journal for Man Machine Studies*, 27:221-234, 1987.
28. Rachlin, J., Kasif, S., Salzberg, S., and Aha, D.W. Towards a better understanding of memory-based reasoning systems. In *Proceedings of the 11th International Conference on Machine Learning,* pp. 242-250, 1994.
29. Rennie, J. D., Shih, L., Teevan, J., and Karger, D. R. Tackling the poor assumptions of Naïve Bayes text classifiers. In *Proceedings of the 20th International Conference on Machine Learning,* pp. 616-623, 2003.
30. Roth, D. Learning in natural language. In *Proceedings of the International Joint Conference of Artificial Intelligence,* pp. 898-904, 1999.
31. Sahami, M. Learning limited dependence Bayesian classifiers. In *Proceedings of the 2nd ACM SIGKDD Conference,* pp. 335-338, 1995.
32. Witten, I. and Frank, E. *Machine Learning Algorithms in Java.* Morgan Kaufmann, 2000.

Preliminary Study of Attention Control Modeling in Complex Skill Training Environments

Heejin Lim[1] and John Yen[2]

[1] Texas A&M University, College Station, TX 77843, USA
hjlim@cs.tamu.edu, jyen@ist.psu.edu
http://people.cs.tamu.edu/hjlim
[2] The Pennsylvania State University, University Park, PA 16802, USA

Abstract. In complex skill-training systems, trainees are required to master multiple skills in a limited time, which may produce a large mental workload. Increased workload often affects performance, and trainees may get distracted or overloaded during training. Attention control is a critical activity in time-sharing environments with automatic tasks, and psychologists found that better attention control strategies can develop through training. Even though attention management is a key skill has to be acquired, it has not been considered to assess as a user model content sufficiently. In this paper, we propose an approach for attention-control modeling by detecting regular behavioral patterns that potentially explain the interdependency between primary and subtask performance. We can detect trainees' attention shift between tasks by interpreting the serial episodes of behaviors that have been uncovered. As a high attention needing training domain, we used Space Fortress game in which continuous input stream of ship maneuvering and intermittent event data are the source of the user model. We found the dependencies between these heterogeneous, multi-time streams and the point of attention shift. Domain experts or training coaches can infer the trainees' attention-control skill based on the detected rules of pattern that help them to instruct desirable strategies to handle multi subtasks.

1 Introduction

A user model consists of all the information and explicit assumptions regarding all aspects of the user. User modeling techniques can effectively support learning on demand by helping users to identify opportunities to learn additional functionality relevant to the task at hand and to prevent people from getting stuck in a sub-optimal plateau. Generally, user models address three issues: (1) inferring the user's knowledge or general abilities; (2) recognizing the user's plans or goals; and (3) predicting the user's inferences and behaviors. The first issue is also called the assessment problem. Assessment has been an important issue because even when it is possible to process numerous observations about a person, with carefully chosen and interpreted references to an extensive empirical database, the task of dealing with the uncertainty associated with the evidence is challenging [10]. The uncertainty of assessment is unavoidable because of the gap between the observed and the inferred evidence and the conclusions drawn. Recently, user modeling has shifted its focus from dialogue systems to interactive applications. When the user controls interactions with applications such as games or training programs, it is hard to identify relevant information that will become the content of the user model and may later on serve to justify an adaptation step.

Knowledge representation and machine learning technique have been used to build user model. Compared to knowledge representation based techniques, machine learning typically accepts observations of user behaviors rather than assumptions that are statements about the user or already formulated in the internal representation. Machine learning techniques implement some sort of uncertainty by using probability, distance measures, and network weights, etc. From the standpoint of acquisition of user-model content and decision tasks, it has an advantage over observing data and making direct decisions from learning algorithms [15]. The purpose of the user model in training programs is to improve the training performance based on the detection of the trainee's current skills and less-than-optimal strategies. In complex skill training, the substantial numbers of individuals fail to develop proficiency and the performance of an expert is qualitatively different from that of a novice. Trainees are required to master multiple skills in a limited time, which may produce a large mental workload. The difficulty is that some trainees can perform an individual component skill well, but cannot operate well in high-workload situations [17]. The attention-control skill is defined as the ability to allocate attention away from an automatic task to concentrate more resources on non-automatic tasks. According to cognitive scientists [22,6], we can perform multiple tasks at one time, as long as we have enough attentional resources and the tasks do not interfere with each other.

In psychological perspective, trainees usually move from explicitly controlled processes into implicit, automatic processes [18,16]. Therefore, the author hypothesizes that trainees' attention control skill can be assessed by modeling automaticity of the primary task, while concurrently manipulating the secondary tasks. In other words, as the skill in the primary task increases, available attention to handle secondary tasks will increase, which enlarges reserve capacity, so that trainees can positively respond to secondary tasks. In this research, we propose an approach for attention-management modeling using Data Mining techniques. We apply rule-discovery methods aimed at finding relationships from the multiple time series with modified episode rules [13]. By detecting regular behavioral patterns that potentially explain the interdependency between primary and subtask performance, we can assess trainees' expertise in attention control and map out the attention shift between tasks.

2 Related Work

Finding out hidden rules that can represent dependency between one time series data and the other one is interesting problem. The association discovery rule

[2] has been developed for finding coexistence of certain values together from a set of transactions. The AprioriAll algorithm [1] can operate on categorical data and considers the coexistence of items within a range. Many applications have used this algorithm to detect association or sequential patterns in single time series data [7,21]. However, the effort to handle multiple, heterogonous time sequences with concurrent events has been tried only rarely. Researchers have tried to discover temporal patterns from multiple streams of homogeneous time series. MSDD (multi-stream dependency detection)[14] method uses a general-to-specific algorithm to detect significant dependencies between multiple streams. It treats dependency detection as an efficient search for the k most predictive rules in search space but it has some limitation to a small number (< 6) of categories in time-series data. Because it is sensitive to time interval it is inappropriate for detecting frequent serial events regardless of intervening events. Höppner's method [9] considers the time interval and the order of events. He tried to discover temporal patterns in a single series of labeled intervals. He mapped the continuous time series into attributed intervals to apply temporal logic to the single time state sequence. In fine-grained domains such as Space Fortress game, most events have relatively short durations that have no significant meaning for intervals, it gets complicated when try to map into time intervals. Das et al. have transformed continuous time series into tractable, discrete sequences by using clustered windows [3]. They extended their framework for two series by merging them into a single, discrete sequence. In their work, they did not sufficiently address the problem of how to order clustered data from different time streams into one sequence when the temporal order should be considered.

As another approach to find association of behavioral patterns, Subramanian et al. studied in tracking the evolution of human subjects' control policies from a large, sequential corpus of low-level data [20]. They partitioned episodes into nearly stationary segments using Kullback-Leibler (KL) divergences to track when the change in policy between adjacent segments is significant. The learning model of control policies is a lookup table that contains a distribution of the actions that were taken in response to each observed perceptual input vector in the episodes. However, the lookup table model needs a more compact representation of the detected model that can be conveyed to human trainees.

For multimodal human-computer interaction, inferring the focus of human attention has been recognized recently as an important factor for the effective communication and collaboration between human and computer. Horvitz et al. tried to model human attention such as sensing and reason [8]. They employed a Bayesian network (BN) for inferring attention from multiple streams of information, and for leveraging this information in decision-making under uncertainty, such as an attention-sensitive alerting system. They modeled attentional focus in a single period with many nodes that link to a focus variable with probabilistic value. One limitation of BN is that the system designer should define variables that possibly affect the attentional focus. However, if the network is densely connected, then inference in the network is intractable. In [19] an approach for modeling attention focus of participants in a meeting via an hidden Markov

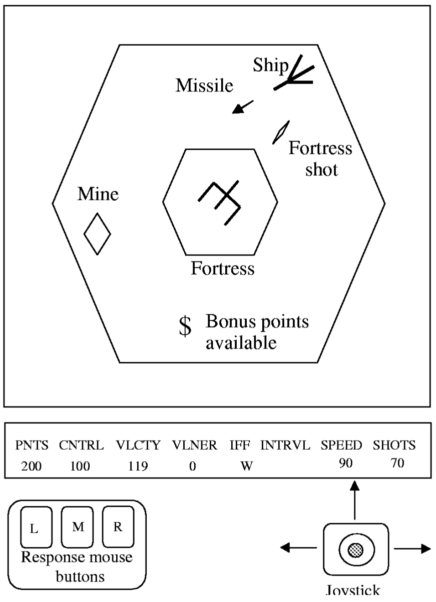


Fig. 1. A Schematic diagram of the Space Fortress game

model was presented. They detected and tracked all participants around the table, estimated their gaze direction by neural network, and mapped the observed gaze to the likely target using a probabilistic framework.

3 Problem Domain: Space Fortress

The Space Fortress (SF) game was developed at the Cognitive Psychophysiology Laboratory of the University of Illinois to simulate a complex and dynamic operational environment [12]. This paper used the ongoing version for distributed team-training SF game as shown in Fig.1. SF game requires a highly concurrent and coordinated use of perceptual and motor skills. Positive transfer from SF training to flight training for U.S. Army helicopter pilots and Israeli Air Force pilots supports the representative nature of the version developed by Gopher et al. [5]. Based on Frederikson & Whites' analysis [4], the SF task can be divided into three goals: (1) circumnavigating the fortress while destroying it as often as possible, (2) handling mines properly, and (3) gaining bonuses. These three goals must be achieved for a high total score. The total score is based on the summation of four sub-scores: Speed (how fast the ship response to friend and foe mine respectively) , Velocity (how low can the ship speed maintained), Points (how many times the ship destroyed fortress and mine or damaged by fortress and mines), and Control (how well the ship maneuvered between the two hexagons). We can consider that the primary task is to navigate between two hexagons at low speeds and the secondary tasks are to handle mines, to earn bonuses, and to destroy the fortress. Our approach separates destroying the fortress from the primary goal because we want to classify subtasks according to their relative attentional component.

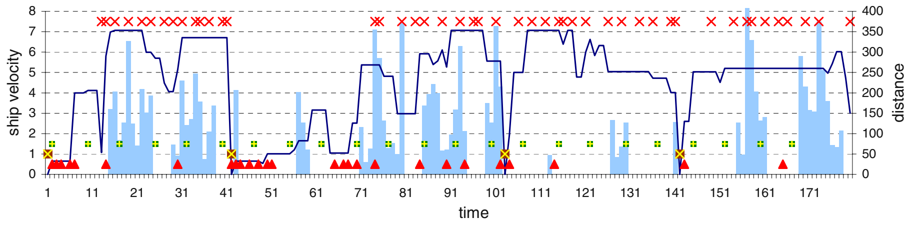
Because the SF system accepts concurrent motor actions from various input devices, it is hard to obtain each skill assessment from multiple continuous stream of performance that shows significant fluctuations over time. We traced the changing of domain states and occurrence of events by generating a history file that contains a long sequence of time-indexed data sets of mixed continuous time series and various other types of events. And then extracted the sequence of skill performance data to find out whether the alteration of primary task performance is related to other subtasks because the performance changing patterns can provide important information about the trainees' attention shift. To see the changing of primary skill performance, i.e., ship velocity, corresponding to secondary skills, let us examine examples of a contingency analysis of SF. Fig.2 illustrates ship velocity change, distance between ship and mine when mine appears, appearance of bonus related characters, ship re-positioning, fortress shell creation, and ship's wrapping events. In this paper, we only consider interdependency between ship velocity skill and mine existence. Fig.2(a) shows that subjects with low level skill showed high ship speed during trials. We can observe that, even when there is no mine, ship speed was still very high because the subject has such a low skill of ship control that the existence of mine did not affect the change of speed significantly. Fig.2(b) shows an example of moderately skilled subjects. Notice that this contingency highly seems that the speed rises rapidly when mines appear and significantly decreases after mine. We find that speed fluctuation is more remarkable than low skilled subjects. Fig.2(c) shows the result from highly skilled subjects. In general, they display low ship speed during trials even when mines appear. Moreover, we can find some regularity that explains dependencies between primary and secondary tasks, as well as individual differences. We can glean these valuable hidden rules from multiple time streams by data mining. For example, a discovered dependency rule is maybe "appearance of mine is followed by increasing of ship velocity with confidence 0.5".

4 Attention Control Model

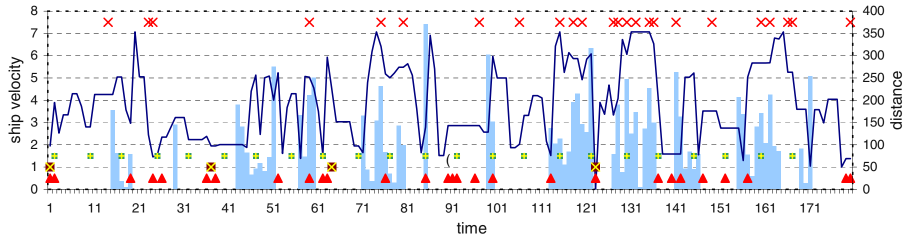
4.1 General Framework

Given multiple time series of heterogeneous types, an attention control model can be constructed. Fig.3 shows an example framework. The following procedure provides the overall sequence, and each process is described in detail below:

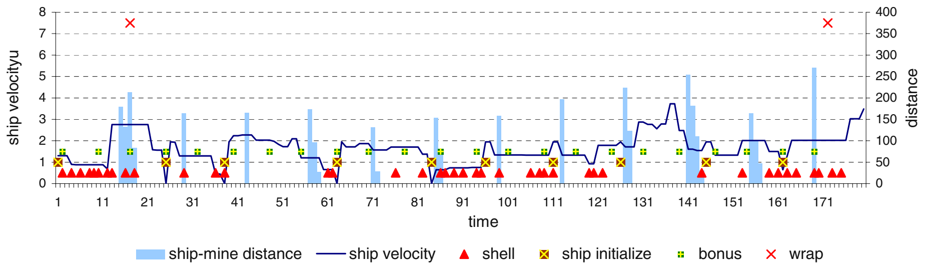
1. Transform the continuous time series into a sequence of categorical data.
2. Transform occasional event tuples into an event sequence.
3. Transform multiple-event sequences into single-event sequences with concurrent event mapping.
4. Apply a serial-episode detection algorithm.



(a) Velocity score: -658



(b) Velocity score: -294



(c) Velocity score: 1272 (time span = 1.0 sec., # of time series = 180)

Fig. 2. Contingency examples of Space Fortress game

5. Generate rules from detected episodes.
6. Filter the generated rules with informativeness measurement.

Transforming Continuous Time Series into a Sequence of Categorical Data

Through transformation of continuous time series into a sequence of qualitative data, we can extract knowledge from a time series that has fluctuations. We can reduce the continuous time-series data into a compact sequence of knowledge including a primitive shape of the pattern. We segmented the time series into subsequences using a *sliding time window* then clustered these subsequences using a suitable measure of pattern similarity method, e.g., K-means clustering, and then assigned categorical symbols to the fragmented subsequences that correspond to the clustered value.

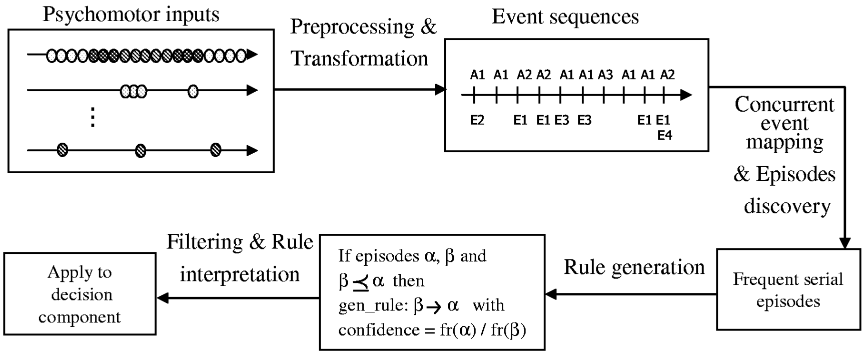
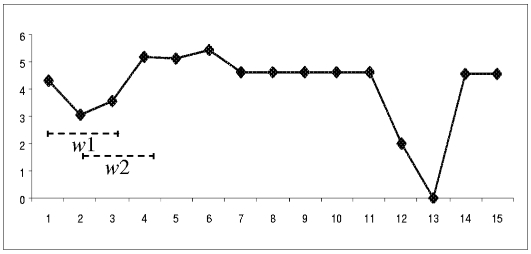

Psychomotor inputs
Preprocessing & Transformation
Event sequences
A1 A1 A2 A2 A1 A1 A3 A1 A1 A2
E2 E1 E1 E3 E3 E1 E1 E4
Concurrent event mapping & Episodes discovery
Frequent serial episodes
Rule generation
If episodes α, β and β ≼ α then gen_rule: β → α with confidence = fr(α) / fr(β)
Filtering & Rule interpretation
Apply to decision component

Fig. 3. General framework for modeling attention control



Clusters	Centroids
A	1.8833
B	1.1059
C	0.6082
D	-1.5818
E	-0.0941

sliding time window (w=3)
#windows = n – w +1 (15-3+1=13)

Time index	< t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15} >
S	< 4.30	3.07	3.54	5.18	5.11	5.44	4.60	4.60	4.60	4.60	4.60	1.99	0	4.54	4.54 >
TS	< e	b	c	e	e	e	e	e	e	d	d	b	a >		

Fig. 4. Transformation of continuous time series by clustering

These steps provide a useful method for discretizing time series [3] and bring up several related issues. First, how should the width of the sliding time window be decided? The window size depends on the time scale the user is interested in. A simple strategy for determinizing width w is that we do not consider the size at first, but after detecting and interpreting rules, adjust the size according to the informativeness of extracted rules decided by domain experts. Here, we set $w = 3$ after comparing results of various window size; it reflects well any change of velocity with a fine-grained time interval. Second, how should the time series data in a window be represented for further analysis? In this study, the change of speed is the main knowledge to trace so that we define the distance function as the slope of speed change between the start point and the end point of the window. For example, a subsequence $< x_i, x_{i+1}, \cdots, x_{i+w-1} >$ has slope $s_i = \frac{x_{i+w-1} - x_i}{w-1}$ and the distance function between subsequence x and y is $d(x, y) = \sqrt{(s_x - s_y)^2}$. By using the degree of change of speed as distance functions, even if average velocity values differ from subject to subject, we still cluster the change of speed in a user-adaptive way. The third issue is, if we use a clustering method to process the data, how many clusters would be reasonable, e.g., choose the value of k for K-means clustering algorithm. We chose five with expectation to group into rapid increasing, slow increasing, no change, slow decreasing and rapid decreasing of velocity.

We have time series data X collected at every time step t with time span ts, such as $X(t) = \{t_i | i = 1, 2, \cdots, n\}$ where n is a time index and $t_{i+1} - t_i = ts$. The value of x collected at t_i varies along the time line. A sliding time window of width $|w|$ on time series X is a contiguous subsequence $s_i = \{x(t_i), x(t_{i+1}), \cdots, x(t_{i+w-1})\}$. From a sequence of X, we get the set of subsequences, s_1, s_{n-w+1} of width w, when window slides left to right by one unit (i.e., window movement $v = 1$). The subsequences of continuous time series are grouped by five centroids as shown in Fig.4 when we set $k = 5$.

Transforming Occasional Events into an Event Sequence

We have several event types, which occur intermittently and concurrently, such as mouse inputs to take bonus and joystick inputs to shoot mine while continuously manipulating joystick to fly ship. Consider k event types $E = \{e_i | i = 1, \cdots, k\}$. For a given event type $e_i \in E, e_i$ has a time step t_i, the time when the event occurred. An event of type e_i can be represented as a pair $(e_i \in E, t_i)$. For example, an event e_1 is input from mouse device by clicking the left or middle button. Let ES_1 is a discrete event sequence for e_1, which represent e_1, which occurred at time stamp t_i. By using the same sliding time window as in the previous continuous time series, the event sequence ES_1 can be transformed to the modified event sequence MSE_1, where $|MES_1| = n - w + 1$. Being different from transformation of contiguous time series, we do not need a clustering procedure. That is, if event e_i occurred in the middle of subsequence (i.e., $x(t_{\frac{i+w}{2}}) = 1$, when $s_i = \{x(t_i), \cdots, x(t_{i+w-1})\}$) then assign the symbol e_i to the subsequence; else assign 0. This process transform the original event sequence into an equal number of data sequences as that of the transformed categorical sequence.

Discovery of Interdependency Episodes

We can detect the frequent combination of events by modifying episode rule [13]. First, in the interest of having information about concurrent events, we convert a set of events on the same time stamp into a new event type as shown in Fig.5. Thus, given n types of events and k clusters of continuous data we may attain at most $k \times \sum_i {}_nC_i = k \times (2^n - 1)$ concurrent events type. An episode is a collection of events in a particular order occurring within a given time window. There is a combinatorial explosion problem when n gets larger; however in training domains, we usually have few subtasks to handle. Thus, we do not investigate the combinatorial issue here. Because events from different sequences contain performance information for the corresponding task, the event set that occurs at the same point in time should be considered as a specific event that requires the trainees attention control. A modified serial-episode detection algorithm is a promising method when the order of events in a fine-grained

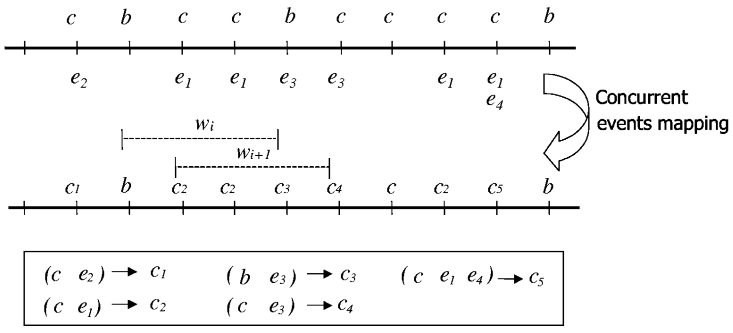


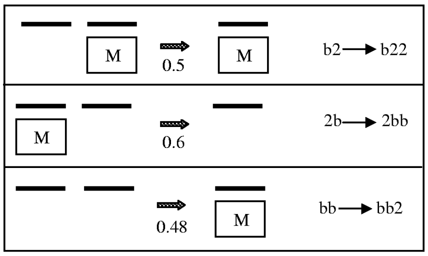
Fig. 5. The example event sequence and two windows

time period has significant meaning. From parallel episodes, serial episodes are detected by recognizing the order of events in windows by screening with a minimum frequency (i.e., min_frequency). This is useful when the size of the sequence is small so that the scanning time is trivial for recognition.

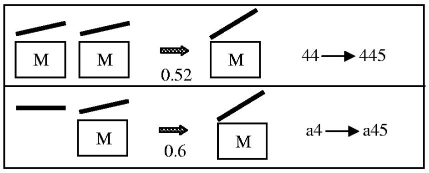
```
//Serial episode detection algorithm
Compute $C_1 = \{\alpha \in \epsilon \mid |\alpha| = 1\}$
$l = 1$
While $C_l \neq \varnothing$
//compute frequent episodes
    $F_l = \{\alpha \in C_l \mid freq(\alpha, s, win) \geq min\_freq\}$
    $l = l + 1$
    $C_l = \left\{\alpha \in \epsilon \mid |\alpha| = 1, \forall\beta\beta \prec \alpha, |\beta| < l, \beta \in F_{|\beta|}\right\}$
For all $l$ do
// find serial episode from $F_l$
    $FS_l = \{\alpha \in F_l \mid freq(\alpha_i, s, win) \geq min\_freq, i \leq l!\}$
For all $l, l \geq 2$ do
    For all $j, j \geq l + 1$ do
      If $\beta \prec \alpha$ where $\beta \in FS_l, \alpha \in FS_j, \frac{freq(\alpha)}{freq(\beta)} \geq min\_conf$
       // generate serial rules
         $\alpha \rightarrow \beta$
```

Generating Rules from Episodes

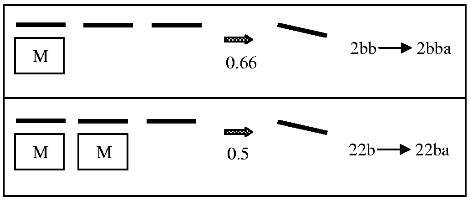
Rule-generating algorithm filters uncovered sequential episodes into serial rules. An objective informative measure, confidence, is used to filter out uninformative rules from the generated rules, providing uncertainty of assessment with statistical methods. This basic approach does not consider any domain knowledge, and as a result, it can generate many irrelevant rules or miss important rules. In Klemettinen et al. [11], rule templates specifying the allowable attribute values are used to post-process the discovered rules. Srikant, Vu, and Agrawal [19] used



(a) low-level skill



(b) moderate-level skill



(c) high-level skill

level / centroid	low	moderate	high
a	-3.3057	-0.0824	-0.3927
b	0.0049	-2.1607	-0.0465
c	1.8532	-0.9993	-1.6302
d	0.9145	0.6355	0.1367
e	-1.0022	1.6109	0.5547

- episode window width = 6
- parallel_minfreq = 0.20
- serial_minfreq = 0.10
- min_conf= 0.40

For low skilled subject's case (a), three serial episodes were detected. As shown in centroid-level table (right below), 'b' represents almost no changing of ship velocity. '2' represents a concurrent event in which behavior 'b' and appearance of mine occurred together. The rule $b2 \rightarrow b22$ explains that in a given time window (6 second in here), if the ship velocity does not change in a while and then mine appears, we can expect that the ship will be in the same velocity with mine appearance in the next step with 50% confidence. (b) and (c) shows detected rules for moderate and high skilled subjects with the same manner as (a).

Fig. 6. Serial episode rule examples

boolean expressions over the attribute values as item constraints during rule discovery. As a second filter, training coaches or experts can detect some useful rules from previously filtered rules.

Fig.6 illustrates the largest serial rules extracted from the data of the three trainees shown in Fig.2. For instance, the first discovered association rule in Fig.6(a) shows that when we observe event *b* and 2 sequentially for the low skilled subject then we expect event 2 to occur, etc. Notice that other events can intervene within these frequent events in the given window. Rules in Fig.6(a) tell us that before the appearance of mine and after disappearance of mine there are rarely change of speed. We can hypothesize that the subject did not shift attention to mine because he/she cannot allocate attention away from primary component that is not yet automated. In Fig.6(b), the rules extracted from the data of moderate-skilled subject show that speed change increases after mine occurs. This implies the mine event distracted the subject because the subject paid attention to mine handling. However, due to limited automation of the primary component (i.e., navigation), the subject's performance for the component is degraded. Fig.6 (c) shows the rules extracted from the data of high skilled subjects. The rules indicate that the speed of ship tends to decrease a little after mine appearance. This may suggest that the subject's speed control skill is well automated, and hence is not disturbed by the appearance of mine.

4.2 Extensions

The ship-mine distance plot in Fig.2 can be used to extract some high-level behaviors in the context of the game such as let the mine approaches the ship closer and then destroy the mine when it is close enough without changing the speed of the ship, or navigate the ship away from the mine to avoid it. The proposed approach can model these behavior patterns by adding a preprocessing task such as calculating the distance between ship and mine for every time step to get a continuous time series. Through transformation, we can get an event sequence that includes distance-changing information and then apply a dependency-detection algorithm to uncover the dependency rules between ship velocity and the ship-mine distance sequence. One challenging issue is applying these rules to the other user-modeling components for coaching purpose. A promising approach is to feed the rules to the systems decision component to generate feedback. By formalizing part of the rules, a KR-based decision component could generate user-adaptive feedback such as directing the user's attention toward certain events, if the events seem significantly linked to performance, and the user does not respond in the right way. However, this should be considered on the premise that training experts or cognitive psychologists analyze the discovered patterns and rules.

5 Conclusion

In complex skill training environment, the degree of users' attention control skill is one of the most important assessments that should be modeled in an appropriate manner. We developed a preliminary computational framework for modeling attention-management as new user-model content by presenting a dependency model capturing the dependency relationship between the trainee's performances in handling multiple skill components. Inspired by the successes of association detection technique of data mining area, we used a modified algorithm to process raw time-series data and discover hidden rules from multiple streams. The traced data are time-indexed streams of psychomotor inputs received from multiple devices and state information resulting from trainees' behaviors. Provided that continuous, multiple-time series exists and can be correctly uncovered, the dependency rules will trace attention shift between skill components. Furthermore, concerning performance fluctuations, serial episode rules will explain to some extent which events cause or follow performance change. For the further study, not only for an individual trial, we need to model the changing of attention control capability as trials go on. The increasing rate of attention control skill would be a useful user model content.

Acknowledgements. The authors would like to thank Richard A. Volz and Wayne Shebilske for helpful discussion. We would also like to thank Yoonsuck Choe for his helpful comments. This research was supported by AFOSR MURI.

References

[1] Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., and Verkamo, A. I.: Fast Discovery of Association Rules. Advances in Knowledge Discovery and Data Mining, MIT Press Chap. 12 (1996) 307–328

[2] Agrawal, R., Imielinski, T., and Swami, A.: Mining association rules between sets if items in large databases. In: Buneman P., and Jajodia, S.(eds.): Proceedings of ACM SIGMOD Conference on Management of Data Washington D.C., USA:ACM (1993) 207–216

[3] Das, G., Lin, K., Mannila, H., Renganathan, G., and Smyth, P.: Rule discovery from time series. Int. Conf. KDDM (1998)

[4] Frederiksen, J.R., White, B.Y.: Principled task decomposition. Acta Psychologica 71 (1989) 89–146

[5] Gopher, D., Weil, M., and Bareket, T.: Transfer of Skill from a Computer Game Trainer to Flight. Human Factors 36(3) (1994) 387–405

[6] Gopher, D, Weil, M., and Siegel, D: Practice under changing priorities: An interactionist perspective. Acta Psychologica 71 (1989) 147–178

[7] Harms, S., Li, D., Deogun, J., and Tadesse, T.: Efficient Rule Discovery in a Geo-Spatial Desicion Support System. Proceedings of the Second National Conference on Digital Government (2002) 235–241

[8] Horvitz, E., Kadie,C.M., Paek, T., and Hovel, D.: Models of Attention in Computing and Communications: From Principles to Applications. Communications of the ACM 46(3) (2003) 52–59

[9] Höppner, F.: Discovery of Temporal Patterns Learning Rules about the Qualitative Behaviour of Time Series. Lecture Notes in Artificial Intelligence 2168. Springer Freiburg, Germany (2001) 192–203

[10] Jameson, A.: Numerical Uncertainty Management in User and Student Modeling: An Overview of Systems and Issues. User Modeling and User-Adapted Interaction 5 (1996) 193–251

[11] Klemettinen, M., Mannila, M., Ronkainen, P., Toivohen, N., and Verkamo, A. I.: Finding interesting rules from large sets of discovered association rules. In Proc. 3rd Int'l Conf. on Information and Knowledge Management, Gaithersburg, Maryland (1994) 401–408

[12] Mané, A., Coles, G. H., Wickens, C. D., and Donchin, E.: The use of additive factors methodology in the analysis of skill. Proceedings of the Human Factors Society-27th Annual Meeting (1983)

[13] Mannila, H., Toivinen, H., and Verkami, A.I.: Discovery of frequent episodes in event sequence. Data Mining and Knowledge Discovery 1(3) (1997) 259 – 289

[14] Oates, T., Firoiu, L., and Cohen, P.R.: Clustering time series with hidden Markov models and dynamic time warping. IJCAI-99 Workshop on Neural, Symbolic and Reinforcement Learning Methods for Sequence Learning (1999)

[15] Pohl, W. and Nick, A.: Machine Learning and Knowledge Representation in the LaboUr Approach to User Modeling. Proceedings of the 7th International Conference on User Modeling. Banff, Canada (1999) 188–197

[16] Rasmussen, J.: Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering. New York : North-Holland (1983) 136–140
[17] Schneider, w.: Training high performance skills: Fallacies and guidelines. Human Factors 27 (1985) 285–300
[18] Shebilske, W. L., Goettl, B. P., and Regian, J. W.: Individual and group protocols for training complex skills in laboratory and applied settings. In: D. Gopher & A. Koriat (eds.): Attention and Performance XVII: Cognitive regulation of performance: Interaction of theory and application. Cambridge, MA: MIT Press (1999) 401–426
[19] Srikant, R.,Vu, Q., and Agrawal, R.: Mining association rules with item constraints. In Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining. Newport Beach, California, AAAI Press (1997) 67–73
[20] Subramanian, D., and Siruguri, S.: Tracking the evolution of learning on a visual-motor task. Technical Report TR02-401, Department of Computer Science Rice University (2002)
[21] Sun, R. and Peterson, T.: Autonomous learning of sequential tasks: Experiments and analysis. IEEE Transactions on Neural Networks 9(6) (1998) 1217–1234
[22] Wickens, C. D. and Hollands, J.: In: C. Wickens and J. Hollands(eds.):Engineering Psychology and Human Performance Prentice Hall, Chapter 11 (2000) 439–479

The Reconstruction of the Interleaved Sessions from a Server Log

John Zhong Lei and Ali Ghorbani

Faculty of Computer Science
University of New Brunswick
Fredericton, NB, E3B 5A3, Canada
{john.lei,ghorbani}@unb.ca

Abstract. Session reconstruction is an essential step in Web usage mining. The quality of reconstructed sessions affects the result of Web usage mining. This paper presents a new approach of reconstructing sessions from Web server logs using the Markov chain model combined with a competitive algorithm. The proposed approach has the ability to reconstruct interleaved sessions from server logs. It is robust even if the client IP is not available in the log file. This capability makes our work distinct from other session reconstruction methods. The experiments show that our approach provides a significant improvement in regarding interleaved sessions compared to the traditional methods.

1 Introduction

Web usage mining is a critical process for analyzing users' browsing behavior in adaptive Web system [4]. The mining data can be collected from various sources. The most important source for performing Web usage mining is the Web server log. However, the server log is a massive file that records the browsing history of all users and that usually contains a large amount of irrelevant information and noise. Obviously, refining these data can improve the quality of any analysis of it. Therefore, collecting reliable data is the foundation of Web usage mining [5]. In an attempt to collect reliable data, a number of issues in Web usage mining must be processed before the mining algorithms can be applied. These include developing a model of the Web log data, developing techniques to clean the raw data to eliminate outliers and irrelevant items, grouping individual page accesses into semantic units, and specializing the generic data mining algorithm to take advantage of the specific nature of the Web log data. In this preprocessing stage, one of the key steps in this preprocessing stage is to reconstruct sessions from the raw Web server log for the mining tasks.

The existing session detection methods usually consider the server log as a serial sequence of sessions. The assumption of these approaches is that a session starts only after another session finishes. However, in practice the sessions usually appear in an interleaved way. It happens in many scenarios of Web browsing. For instance, a user's performing multiple tasks simultaneously will produce interleaved sessions; multi-users through a proxy server will also result in interleaved sessions even though each user is performing a mono-task. If there were a large number of interleaved sessions in the server log, the performance of these approaches would dramatically decrease. This situation could be even worse when the user's IP address is not available.

In this paper, we propose a new method of reconstructing sessions from the server logs for Web usage mining. The aim of our approach is to solve the problem of interleaved sessions so that the system can be robust even when the user's IP address is not available in the Web logs. The proposed approach consists of two steps. In the first step, interleaved sessions are separated and put into a session stack. The session stack maintains an index of current active sessions. All active sessions are stored in a session stack. The task of the second step is to detect the end of an active session. When the end of a session is detected, its associated active session will be closed.

The rest of the paper is organized as follows. In the next section, we briefly review the related work. Section 3 presents the proposed approach. The experiments to evaluate the performance of our approach are described in Section 4. Finally, Section 5 gives a summary and concludes the present study.

2 Related Work

Identifying Web users and user sessions is a complicated task in the data preprocessing phase of Web mining [2,5]. Several approaches have been applied to session reconstruction. The naïve but commonly used approach is simply to generate sessions from a log by dividing this log into equal-length or equal-duration subsequences. Three heuristics for session identification are described in [8]: session duration heuristic (H1), page-stay time heuristic (H2), and referrer-based heuristic (Href). H1 and H2 are based on the assumption that the time a user spends on a session or a page correlates to whether the session ends. H1 and H2 roughly cut the request sequence into sessions by a predetermined timeout threshold. Both H1 and H2 may cause many errors because there is no guarantee that all users have the same time-limited behaviors. In the reference-based approach, a new session is created when a page is not referred by the pages in the previous session. The main problem of this approach is that the referral page is not always available. If a user typed the Web address instead of clicking a link, the referral page could not be obtained. The Href approach usually cuts a session into several short sessions.

The *n-gram* language model is used in [3] to identify session boundaries. In this work, an *n-gram* language model, which has been widely used in statistical language modeling for speech recognition, was applied to identify session boundaries. The assumption of this approach is that a significant decrease of the probability of a browsing page sequence indicates a session boundary. This approach performed better than the time-oriented heuristic. However, in their approach, the problem of interleaved sessions is expected to be solved in advance based on the user IP address. When the IP address is not reliable to identify a user, the performance of this approach will decrease.

The Markov chain model have recently been used to model user navigational behaviors in the World Wide Web [6,10,11]. The Markov chain model assumes that a user will click a link following the pattern which depends on his/her previous browsing pages. In particular, an m-step Markov chain model assumes that a user's next page is determined by the last m pages visited by him/her. A linear interpolation of the Markov model is used in [11]. This model reduces the effect of sparse problem that is caused by insufficient training data. Moreover, it is easier to implement and requests less computation time compared to the *n-gram* model.

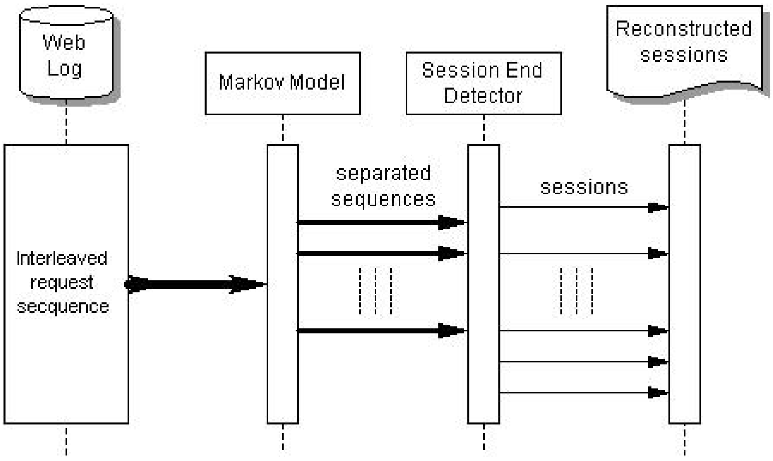


Fig. 1. Session Reconstruction Model

3 Session Reconstruction

In an effort to reconstruct the interleaved sessions from the Web logs, we must first separate the sessions and then detect the end of the sessions. This process includes two stages:

1. *Construct*: build sessions by identifying which active session a requested page should belong to and
2. *Close*: determine whether an active session should be closed or not by detecting the end of the session.

Our reconstruction model is shown in Figure 1. In this approach, we use a session stack to store and to maintain an index of the active sessions. In the first stage, the users' requests are presented to the session stack in sequence. We use the Markov chain model to detect which active session a request should belong to by comparing the probabilities of a request's adding to these sessions. The session with the highest probability will obtain this request. On the other hand, if the highest probability were lower than a threshold, a new active session would be built, and the request would become the first entry of the new session. For the second stage, we propose two approaches to determine whether a session should be closed or kept open. The first approach is to apply the H2 method to detect the end of the sessions. That is, if the time gap between a request and the last request of a session were greater than a threshold, this session would be closed. The second approach is based on a competitive algorithm. We assign an "energy" value to each active session. The "energy" of a session increases when it wins a new request and decreases when it loses. An energy threshold is set to detect the end of the active sessions.

3.1 The Isolation of the Interleaved Sessions

As illustrated above, we use a session stack to maintain an index of the active sessions and a vector to store the corresponding probabilities. These probabilities show the likelihood that a request should be added to the active sessions.

The Markov chain model is used to perform this task. The user's browsing sequence is considered as a series of transition changes. In particular, the m-step Markov chain model assumes that a user's next step depends only on the last m pages he or she browsed. The probability of the next page can be calculated by the following equation:

$$P(r_{n+1}|r_n, r_{n-1}, ..., r_1) = P(r_{n+1}|r_n, r_{n-1}, ..., r_{n-m+1})$$

where r_i is the ith page request in a session; r_n is the current request, and r_{n+1} is the next request.

To reduce the effect of the sparse problem, the linear interpolation of Markov model is used in our approach. The Markov models are combined by using a weighted sum of the available models as the following equation:

$$P = w_1 \times P(r_{n+1}, r_n) + w_2 \times P(r_{n+1}, , r_{n-1}) + \cdots + w_m \times P(r_{n+1}, ..., r_{n-m+1})$$

where r_i is the ith page request in a session; $w_1, w_2, \cdots, w_m$ are the weights assigned to the history pages. The weights indicate the level of influence that the history pages have on the subsequent page.

According to the definition, in a one-step Markov chain model, a user's next step depends only on the last page he or she browsed. The transition matrix is used to map the probabilities of the state's transforming. A one-step probability transition matrix Q is an $n \times n$ matrix, which represents the one-step transition probability between any two pages. In the matrix, the row i contains the transition probabilities from page i to all of the other pages in one step. The column j contains the transition probabilities from the other pages to page j in one step. The transition matrix is calculated from the training data by the following equation:

$$Q(i, j) = \frac{(r_i r_j)}{(r_i)}$$

where $(r_i r_j)$ is the occurrences of sequence i, j in the training data, and (r_i) is the occurrences of request i in the training data.

The m-step transition matrix is described by the mth power of Q, Q^m. In an m-step transition matrix, $Q^m(i, j)$ is the probability of moving from page i to page j in m steps.

If there were a session $S = (r_1, r_2, ..., r_{n-m+1}, ..., r_{n-1}, r_n)$, the probability of a page request r_k's belonging to this session could be calculated as follows according to the linear interpolation of Markov model discussed above:

$$P = w_1 \times Q(r_n, r_k) + w_2 \times Q^2(r_{n-1}, r_k) + \cdots + w_m \times Q^m(r_{n-m+1}, r_k)$$

where $w_1, w_2, \cdots, w_m$ are the weights assigned to the history matrix. The values of w's indicate the level of influence that the history matrix has on the subsequent page. Normally, we choose $1 > w_1 > w_2 > \cdots > w_m > 0$, so that the later requested page has more influence on the subsequent page. The values of the optimization parameters need to be found by performing the experiments with different values.

3.2 The Detection of the End of a Session

In the second stage, to determine the end of the sessions, we apply two methods: the time oriented heuristic method and the competitive method.

Time Oriented Heuristic: There are two types of Time Oriented Heuristics [8]: H1 and H2. The H1 method defines the total time spent in a session. It assumes that the total duration of a session is less than a threshold. The H2 method limits the time spent by a user for browsing a page. It assumes that if there were a long elapsing time between two visited pages, the latter page should belong to another session [1,7].

The H2 method has been chosen as one of our approaches to determine the end of sessions. Because the log is in time sequence, each active session in the session stack keeps the time stamp of its latest page. Based on the H2 heuristic, a session should be closed if the elapsing time between the latest page of a session and the consequent page were longer than an H2 threshold.

To simplify the computation, we sort the active sessions in time order. Whenever a page is added to a session, this session will be moved to the end of the active session index. In this way, if a session should be closed by H2 heuristic, all sessions ahead of it should be closed as well. The combined Markov model and H2 algorithm is outlined in Figure 2.

Competitive Method: Another approach for detecting the end of a session in this work is the competitive algorithm. We assign a variable called "energy" to each active session. This variable will be updated in a competitive mode. If a request were added to a session, we would said this session wins the competition. Based on the competitive rule, the winner's energy will be rewarded, and the energy of the other sessions will be reduced. A session will be closed when it loses all of its energy (i.e., the energy value becomes zero). The algorithm of the proposed competitive Markov model is outlined in Figure 3.

Rewards and punishments are essential in a competitive method. We chose two factors to calculate the values of the rewords and of the punishments. One is the number of the current active sessions, and the other is the probability of a request's belonging to a session. They are called Compete and Reward by Sessions (CRS) and Compete and Reward by Probabilities (CRP), respectively.

– *Compete and Reward by sessions (CRS)*: In this approach, the update of a energy variable is based on the number of the current active sessions. If a session won a page, its energy would increase by $e^{-\alpha\frac{1}{n}}$, in which α is an amplifying factor, and n is the number of the current active sessions. The winning session would obtain more rewards if there were more active sessions. If a session didn't win a page, its energy would decrease by $e^{-\alpha(1-\frac{1}{n})}$. A session would obtain less punishments if there were more active sessions.
– *Compete and Reward by Probabilities (CRP)*: In this approach, the energy values will be updated according to the probabilities that a request is added to the sessions. If an active session won a page, its energy would increase by $e^{-\alpha(1-P)}$, in which P denotes the probability that the page should be added to the session, and α is a preset amplifying factor. The winning session would obtain more rewards if its probability were higher. On the other hand, if an active session didn't win, its energy would decrease by $e^{-\alpha P}$. This session would get fewer punishments if its probability were higher.

4 Experiments and Evaluation

In this section, we experiment with and evaluate the proposed methods by using the Web server log file of the University of New Brunswick.

4.1 Test Environment

The UNB server log is used as our test data. We collected four days of the UNB server log. After cleaning the data by removing noise and broken links, we collected 22785 sessions and divided them into the training and test parts. The training part contains 20785 sessions with a total of 119259 requests. The test part contains 2000 sessions with a total of 11673 requests. The collected data are composed of 2764 unique pages.

4.2 Experimental Process

The test process of session reconstruction is shown in Figure 4. We created the pseudo real sessions from the raw log file. These sessions are divided into the training set and the test set randomly. The training set is then used to generate the transition matrix, which could be used to extract the request probabilities. The test set is used to generate test log file and is then used to evaluate the reconstruction.

```
Input:
Web log file $l = \{(r_i, \tau_i), i = 1 \ldots k\}$, where $r_i$ is the $i$th request and $\tau_i$ is
the time stamp of the request.
Output:
Reconstructed sessions file.
Initialization:
$\gamma$: time threshold of H2 approach.
$\beta$: probability threshold.
S: session stack, stores active sessions. $S = \{s_1, s_2, \ldots, s_m\}$, where each
session $s_j = \{(r_{j1}, r_{j2}, \ldots, r_{jn}), \tau_j\}$, in which $(r_{j1}, r_{j2}, \ldots, r_{jn})$ is the request
sequence of the session $s_j$, $\tau_j$ is the time stamp of the last request of the
session.
Begin
for $(r_i, \tau_i) \in l$ do
  for $s_j \in S$ do
    if $\tau_i - \tau_j > \gamma$ then
      $output \leftarrow s_j$
      remove $s_j$ from S
    else
      compute $P(s_j r_i | s_j)$
      /* compute the probability of request $r_i$ belongs to session $s_j$ */
    end if
  end for
  $P_{max} = \max P(s_j r_i | s_j)$
  if $P_{max} > \beta$ then
    index=the index of the session with $P_{max}$
    $S_{index} \leftarrow r_i$
    $\tau_{index} = \tau_i$
    move $s_{index}$ to the end of S
  else
    $s_{new} = \{(r_i), \tau_i\}$
    $S \leftarrow s_{new}$
  end if
end for
End
```

Fig. 2. Algorithm: the Markov model combined with the H2 approach

Preprocessing: The raw UNB server log is shown in Figure 5. This log file follows the common log file format defined by W3C [9]. The **"fctn1-1861.nb.aliant.net"** is the client IP address . The following **"- -"** are the remote logname of the user and the username which the user has authenticated himself or herself. These values are not available in our server log. **"06/Jan/2003:00:01:08"** is the time stamp of the request. **"GET"** is the method of the trasaction. **"/registration/"** is the destination URL requested by the client. **"HTTP/1.1"** is the HTTP protocol. **"304"** is the HTTP return code. The following **"-"** is the size in bytes of the response sent to the client. It is 0 bytes in this case. "-" is the referrer, the URL refers to the current request.

```
Input:
Web log file $l = \{(r_i, \tau_i), i = 1 \dots k\}$, where $r_i$ is the $i$th request and $\tau_i$ is
the time stamp of the request.
Output:
Reconstructed sessions file.
Initialization:
$FR()$: reward function.
$FP()$: punish function.
$energy_{ini}$: initial energy.
$\beta$: probability threshold.
$S = \{s_1, s_2, \dots, s_m\}$: session stack, where each session $s_j = \{(r_{j1}, r_{j2}, \dots, r_{jn}), energy_j\}$
consists of the request sequence $(r_{j1}, r_{j2}, \dots, r_{jn})$ and its associated energy
$energy_j$.
Begin
for $(r_i) \in l$ do
  for $s_j \in S$ do
    if $energy_j <= 0$ then
      $output \leftarrow s_j$
      remove $s_j$ from S
    else
      compute $P(s_j r_i | s_j)$
      /* compute the probability of request $r_i$ belongs to session $s_j$ */
    end if
  end for
  $P_{max} = \max P(s_j r_i | s_j)$
  if $P_{max} > \beta$ then
    index=the index of the session with $P_{max}$
    $S_{index} \leftarrow r_i$
    $energy_{index} = energy_{index} + FR()$
    for $(s_j) in S, j \neq index$ do
      $energy_j = energy_j - FP()$
    end for
  else
    $s_{new} = \{(r_i), energy_{ini}\}$
    $S \leftarrow s_{new}$
  end if
end for
End
```

Fig. 3. Algorithm: the Markov model combined with the competitive approach

"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)" is the client's browsing agent.

The pseudo real sessions were generated from the original log for training. First, the noisy requests such as bad links were removed. Second, the real sessions were identified based on the combined information of the IP address, the time stamp, and the referred pages. We also used a variable-threshold time-oriented approach to alleviate the cache and empty-referrer problems. Finally, those sessions that were vague, incomplete, or without referrers were removed. A part of our reconstructed pseudo real sessions is shown Figure 6.

Training and Reconstruction: The main propose of the training process is to generate the probability transition matrix. The probabilities are generated from counting. We identified each unique page with a number. The counting program

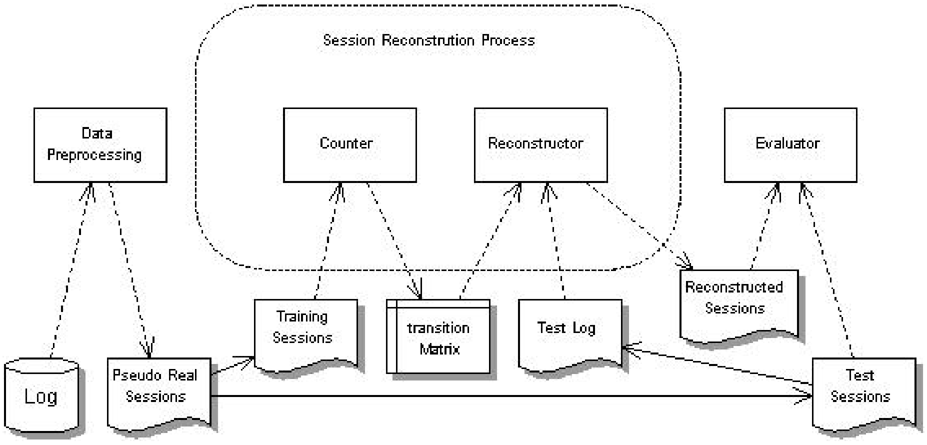
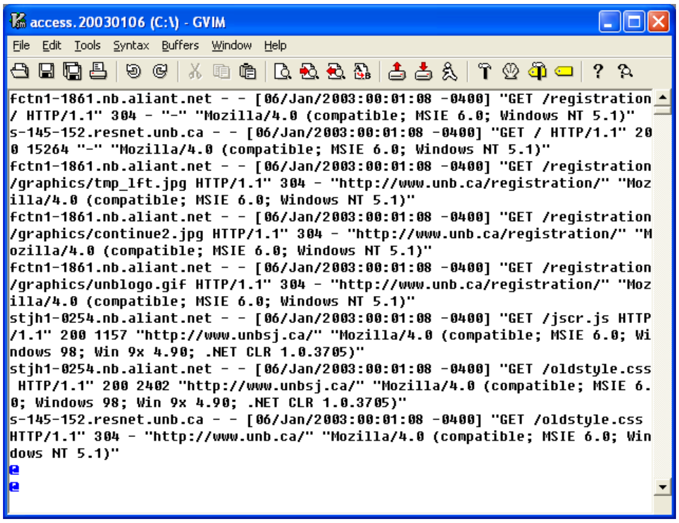


Fig. 4. Experimental Process


access.20030106 (C:\) - GVIM
File Edit Tools Syntax Buffers Window Help

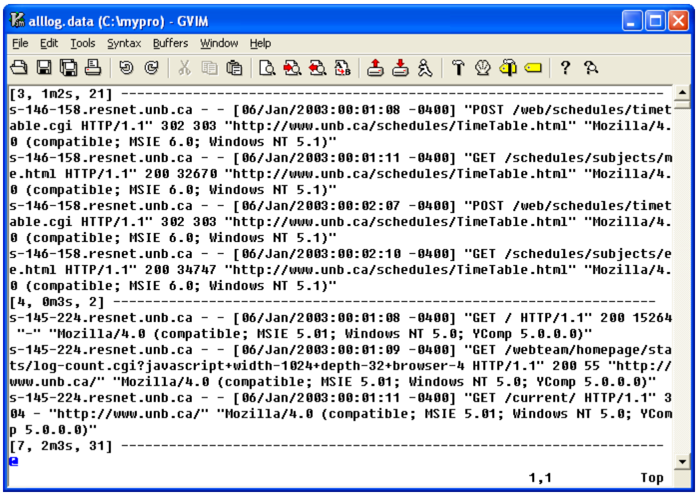
```
fctn1-1861.nb.aliant.net - - [06/Jan/2003:00:01:08 -0400] "GET /registration
/ HTTP/1.1" 304 - "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
s-145-152.resnet.unb.ca - - [06/Jan/2003:00:01:08 -0400] "GET / HTTP/1.1" 20
0 15264 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
fctn1-1861.nb.aliant.net - - [06/Jan/2003:00:01:08 -0400] "GET /registration
/graphics/tmp_lft.jpg HTTP/1.1" 304 - "http://www.unb.ca/registration/" "Moz
illa/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
fctn1-1861.nb.aliant.net - - [06/Jan/2003:00:01:08 -0400] "GET /registration
/graphics/continue2.jpg HTTP/1.1" 304 - "http://www.unb.ca/registration/" "M
ozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
fctn1-1861.nb.aliant.net - - [06/Jan/2003:00:01:08 -0400] "GET /registration
/graphics/unblogo.gif HTTP/1.1" 304 - "http://www.unb.ca/registration/" "Moz
illa/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
stjh1-0254.nb.aliant.net - - [06/Jan/2003:00:01:08 -0400] "GET /jscr.js HTTP
/1.1" 200 1157 "http://www.unbsj.ca/" "Mozilla/4.0 (compatible; MSIE 6.0; Wi
ndows 98; Win 9x 4.90; .NET CLR 1.0.3705)"
stjh1-0254.nb.aliant.net - - [06/Jan/2003:00:01:08 -0400] "GET /oldstyle.css
 HTTP/1.1" 200 2402 "http://www.unbsj.ca/" "Mozilla/4.0 (compatible; MSIE 6.
0; Windows 98; Win 9x 4.90; .NET CLR 1.0.3705)"
s-145-152.resnet.unb.ca - - [06/Jan/2003:00:01:08 -0400] "GET /oldstyle.css
HTTP/1.1" 304 - "http://www.unb.ca/" "Mozilla/4.0 (compatible; MSIE 6.0; Win
dows NT 5.1)"
```

Fig. 5. The raw UNB server log

went through the training sessions and mapped the page transition into a one-step probability transition matrix. This matrix can be used to compute two or more step transition matrixes.

With the transition matrixes, the reconstructor program can reconstruct the sessions by applying the approaches discussed in Section 3. Two methods were used for detecting the end of a session in the experiment: the time oriented heuristic (H2) and the competitive method.

Evaluation: In the test phase, the reconstructed sessions are compared with the test sessions by the evaluator program. If a session in the reconstructed file


alllog.data (C:\mypro) - GVIM
File Edit Tools Syntax Buffers Window Help

```
[3, 1m2s, 21] ---------------------------------------------------------------------
s-146-158.resnet.unb.ca - - [06/Jan/2003:00:01:08 -0400] "POST /web/schedules/timet
able.cgi HTTP/1.1" 302 303 "http://www.unb.ca/schedules/TimeTable.html" "Mozilla/4.
0 (compatible; MSIE 6.0; Windows NT 5.1)"
s-146-158.resnet.unb.ca - - [06/Jan/2003:00:01:11 -0400] "GET /schedules/subjects/m
e.html HTTP/1.1" 200 32670 "http://www.unb.ca/schedules/TimeTable.html" "Mozilla/4.
0 (compatible; MSIE 6.0; Windows NT 5.1)"
s-146-158.resnet.unb.ca - - [06/Jan/2003:00:02:07 -0400] "POST /web/schedules/timet
able.cgi HTTP/1.1" 302 303 "http://www.unb.ca/schedules/TimeTable.html" "Mozilla/4.
0 (compatible; MSIE 6.0; Windows NT 5.1)"
s-146-158.resnet.unb.ca - - [06/Jan/2003:00:02:10 -0400] "GET /schedules/subjects/e
e.html HTTP/1.1" 200 34747 "http://www.unb.ca/schedules/TimeTable.html" "Mozilla/4.
0 (compatible; MSIE 6.0; Windows NT 5.1)"
[4, 0m3s, 2] --------------------------------------------------------------------
s-145-224.resnet.unb.ca - - [06/Jan/2003:00:01:08 -0400] "GET / HTTP/1.1" 200 15264
 "-" "Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0; YComp 5.0.0.0)"
s-145-224.resnet.unb.ca - - [06/Jan/2003:00:01:09 -0400] "GET /webteam/homepage/sta
ts/log-count.cgi?javascript+width-1024+depth-32+browser-4 HTTP/1.1" 200 55 "http://
www.unb.ca/" "Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0; YComp 5.0.0.0)"
s-145-224.resnet.unb.ca - - [06/Jan/2003:00:01:11 -0400] "GET /current/ HTTP/1.1" 3
04 - "http://www.unb.ca/" "Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0; YCom
p 5.0.0.0)"
[7, 2m3s, 31] ---------------------------------------------------------------------

                                                            1,1           Top
```

Fig. 6. The reconstructed pseudo real sessions

perfectly matched a session in the test file, it would be called a "perfect session". If there were only one mismatched request in the reconstructed session, this session would be called a "one-mis session". Both the perfect sessions and the one-mis sessions are considered as "successful sessions" while the sessions with two or more mismatched requests are considered as "failed sessions".

4.3 Experimental Results

We masked the IP address from the test set to simulate a situation in which the IP address is not available. This situation is the extreme case that interleaved sessions happen frequently. We investigated the following approaches:

– **H1**: The standard H1 approach,
– **H2**: The standard H2 approach,
– **Href**: The standard referral heuristic approach,
– **M+H2**: The Markov model combined with the H2 approach,
– **M+CRS**: The Markov model combined with the CRS, and
– **M+CRP**: The Markov model combined with the CRP.

H1, H2 and Href: Both the standard H1 and H2 approaches failed in reconstructing the sessions because most of the sessions are interleaved with each other. Without separating the interleaved sessions, simply using standard H1 and H2 can only roughly cut the request sequence into meaningless pieces. Href also performs poorly as shown in Table 1. The reason is that the interleaved request misled the referral path. If the IP address were not available or the interleaved sessions covered the majority of the Web log, none of H1, H2 and Href could successfully reconstruct sessions.

Table 1. The performance of the Href

	perfect sessions	one-miss sessions	successful sessions
Href	4.3%	40.8%	45.1%

Table 2. The performance of the Markov model combined with the H2 approach

	time threshold	perfect sessions	one-mis sessions	successful sessions
M1+H2	5 min.	30.3%	23.8%	54.1%
M1+H2	**10 min.**	**30.9%**	**24.6%**	**55.5%**
M1+H2	15 min.	30.0%	23.6%	53.6%
M1+H2	20 min.	29.4%	23.1%	52.5%
M1+H2	25 min.	29.0%	22.8%	51.8%

The Markov model combined with the H2 approach: The results of using the 1-step Markove model with the H2 cutoff approach(M1+H2) is shown in Table 2. We conducted the experiments using various time thresholds. The results show that 10 minutes is the best cutoff threshold for our log. According to this result, we tested the 1, 2, 3 step Markov models with 10-minute cutoff threshold of the H2 approach, denoted M1+H2, M2+H2, and M3+H2, respectively. The results in Table 3 show that increasing the number of the steps of Markov model can slightly increase the performance. Compared to simply using the H1, H2 and Href, this result indicates that the Markov model has the ability to separate the interleaved sessions from the Web logs.

Table 3. The performance of various-step Markov model combined with the H2 approach

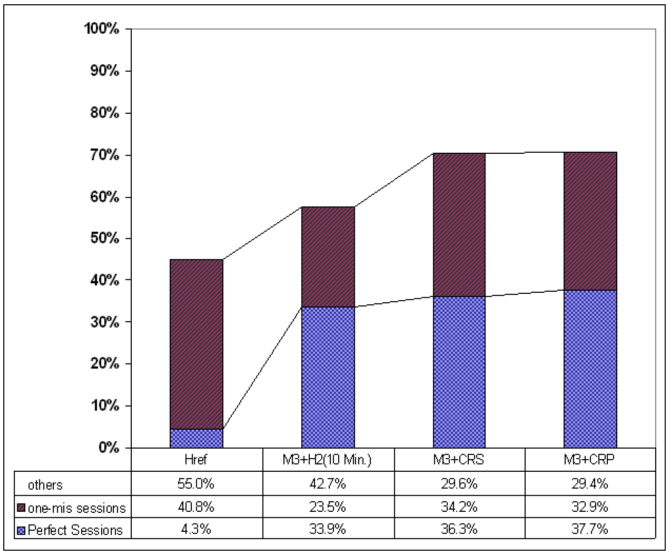
	time threshold	perfect sessions	one-mis sessions	successful sessions
M1+H2	10 min.	30.9%	24.6%	55.5%
M2+H2	10 min.	32.9%	24.3%	57.2%
M3+H2	10 min.	33.8%	23.5%	**57.4%**

Table 4. The performance of the Markov model combined with the competitive approach

	perfect sessions	one-mis sessions	successful sessions
M1+CRS	32.9%	29.2%	62.1%
M2+CRS	35.9%	31.2%	67.1%
M3+CRS	36.3%	34.2%	70.4%
M1+CRP	35.2%	29.1%	64.1%
M2+CRP	36.2%	30.5%	66.7%
M3+CRP	37.7%	32.9%	**70.6%**

The Markov model with the competitive approach: The results of Markov model with the competitive approach are shown in Table 4. The results illustrate that the CRP approach performs better than the CRS approach in the case of perfect sessions. Compared to the time-out cutoff method H2, the competitive approaches significantly improve the performance.

Comparison: The comparison of all of these approaches is shown in Figure 7. Among these approaches, the Markov model combined with the CRP has the best performance. The correct reconstructed session rate increases from 45.1% using the Href to 57.4% using the M3+H2. The competitive algorithm obtains a 13.2% further increase. The correct session rate increases to 70.6%. Compared to the 45.1% of the standard Href, the overall improvement rate is 56.5%.



	Href	M3+H2(10 Min.)	M3+CRS	M3+CRP
others	55.0%	42.7%	29.6%	29.4%
▨ one-mis sessions	40.8%	23.5%	34.2%	32.9%
▨ Perfect Sessions	4.3%	33.9%	36.3%	37.7%

Fig. 7. The performance comparison

5 Conclusion

Session reconstruction is an essential step in Web usage mining. The existing session reconstruction approaches cannot reconstruct the interleaved sessions and perform poorly when the client's IP address is not available. In this paper, we propose a new method of reconstructing interleaved sessions from Web logs. The experiments obtain the promising result that the Markov chain model has the ability to divide interleaved sessions, and can provide a further improvement when it is combined with the competitive approach.

While the existing session reconstruction approaches perform poorly when the IP address is not available or when the interleaved sessions cover the majority of the server log, the proposed approach successfully reconstructs the interleaved sessions from the Web server logs.

Acknowledgments. This work was funded in part through grant RGPIN 227441-00 from the National Science and Engineering Research Council of Canada (NSERC) to Dr. Ali A. Ghorbani.

References

1. Berendt, B., Mobasher, B., Spiliopoulou, M., Wiltshire, J. : Measuring the accuracy of sessionizers for Web usage analysis. In Workshop on Web Mining at the First SIAM International Conference on Data Mining, pages 7-14. April 2001.
2. Colley, R., Mobasher, B., Srivastava, J.: Data Preparation for Mining World Wide Web Browsing Patterns. Knowledge and Information Systems, pages 5-32. 1999.
3. Huang, X., Peng, F., An, A., Schuurmans, D., Cercone, N.: Session Boundary Detection for Association Rule Learning Using n-Gram Language Models. Proceedings of Sixteenth Conference of the Canadian Society for Computational Studies of the Intelligence, pages 237-251. 2003.
4. Kilfoil, M., Ghorbani, A., Xing, W., Lei, Z., Lu, J., Zhang, J., Xu, X.: Toward an Adaptive Web: The State of the Art and Science. the 1st Annual Conference on Communication Networks & Services Research, pages 119–130. 2003.
5. Pitkow, J.: In search of reliable usage data on the WWW. Computer Networks and ISDN Systems, pages 1343–1355. 1997.
6. Sarukkai, R. R.: Link Prediction and Path Analysis Using Markov Chains. Computer Network, pages 377-386. 2000.
7. Spiliopoulou, M., Pohle, C., Faulstich L.: Improving the Effectiveness of a Web Site with Web Usage Mining, WEBKDD, pages 142–162. 1999.
8. Spiliopoulou, M., Mobasher, B., Berendt, B., Nakagawa, M.: A Framework for the Evaluation of Session Reconstruction Heuristics in Web Usage Analysis. INFORMS Journal of Computing, Special Issue on Mining Web-Based Data for E-Business Applications. 2003.
9. Hallam-Baker, P. M., Behlendorf, B.: Extended Log File Format. http://www.w3.org/TR/WD-logfile, accessed: August 19, 2003.
10. Ypma, A., Heskes, T.: Categorization of Web Pages and User Clustering with mixtures of Hidden Markov Models. the International Workshop on Web Knowledge Discovery and Data Mining. July 2002.
11. Zhu, J., Hong, j., Hughes, J. G.: Using Markov models for web site link prediction. Proceedings of the thirteenth ACM conference on Hypertext and hypermedia, pages 169-170. 2002.

On Customizing Evolutionary Learning of Agent Behavior

Jörg Denzinger and Alvin Schur

Department of Computer Science, University of Calgary, Canada
{denzinge,schur}@cpsc.ucalgary.ca

Abstract. The fitness function of an evolutionary algorithm is one of the few possible spots where application knowledge can be made available to the algorithm. But the representation and use of knowledge in the fitness function is rather indirect and therefore not easy to achieve. In this paper, we present several case studies encoding application specific features into fitness functions for learning cooperative behavior of agents, an application that already requires complex and difficult to manipulate fitness functions. Our experiments with different variants of the Pursuit Game show that refining a knowledge feature already in the fitness function usually does not result in much difference in performance, while adding new application knowledge features to the fitness function improves the learning performance significantly.

1 Introduction

Over the last few years, research into learning behavior of agents has intensified, resulting in a large variety of learning concepts for different agent architectures and different application areas. While research into reinforcement-based approaches has focused a lot on possible solutions to the credit assignment problem and consequently the learning function, evolutionary learning approaches covered a large variety of goals related to agent, resp. multi-agent, specific features. These goals included the development of evolutionary algorithms for rather different agent architectures, like the use of genetic programming to learn real agent control programs (see [8]), co-evolutionary approaches on the level of nodes in neural networks (see [10]), or genetic algorithms for sets to learn prototypical situation-action pairs that are used together with the nearest-neighbor rule for action selection (see [3]).

Additionally, other research looked at how various kinds of knowledge can be incorporated into the learning process, ranging from using knowledge that allows a hierarchical learning approach (see [12]), to knowledge defining tasks of growing complexity (see [7]), to transferring knowledge from one agent to another with different abilities (see [2]). Also the ability to communicate and in fact to learn communications to express appropriate knowledge has been explored (see [9]). But one of the key ways to include knowledge into an evolutionary search has not been looked at in more detail: the fitness function. The reported research either just stated a fitness function that could do the job, or the researchers used very general fitness measure ideas that can be instantiated rather differently and then provided a single instantiation. While outside of learning of behavior there has been quite some research into fitness functions (see [6] for a rather general method), for learning of cooperative behavior there is still little advice available for incorporating application specific knowledge into the fitness function.

In this paper, we present case studies regarding adding more knowledge into the fitness function used within the OLEMAS system (see [4]) that has as application domain a huge variety of Pursuit Games (see [3]). The fitness function in OLEMAS started out with the goal to be applicable for all game variants OLEMAS provided. However, learning complex behaviour was slow. Previous work concentrated on evaluating the influence of the control parameters (see [5]) or on improving the genetic operators in certain team scenarios (see [2]). But these works still showed room for improvements.

A lot of the problems that were observed deal with not representing certain knowledge, resp. features, specific to the game variant: orientation of agents, the blocking effect of obstacles and other agents or the particulars of the goal of the game. Our experiments show that representing such key game variant features in the fitness function improves the learning performance compared to a standard general purpose fitness function. While this seems to indicate that learning of cooperative behavior for a human developer just transfers the task of developing good cooperating strategies into the –more obscure– task of developing a fitness function containing the appropriate guiding knowledge, we can also report that the way feature knowledge is represented does not produce big changes, as demonstrated by exploring different ways to measure the base distance between two points. And the knowledge we integrate into the fitness in our case studies still applies to a rather large set of game variants, so that the gains by its use are wide spread.

2 Learning with SAPs and the NN Rule

In this section, we present our evolutionary learning method. To do this, we first present the agent architecture we are using and then we discuss learning using a GA on sets.

2.1 Our Agent Architecture

In general, an agent can be seen as a triple (Sit,Act,Dat) with a set Sit of situations, a set Act of actions, and a set Dat of values of internal data areas. The agent uses its perception of the actual situation and the values of the internal data areas to decide upon the next action. Therefore an agent $\mathcal{A}g$ can be seen as a function $\mathrm{f}_{\mathcal{A}g}$: $Sit \times Dat \to Act$. For an agent in a multi-agent system we can divide the sets into parts concerning the agent itself and parts concerning the other agents. More formally, an element s of Sit has the form $s = s_{Env}s_{Ag}$, where s_{Env} describes the environment the agent acts in without any information about other agents and s_{Ag} provides this information about other agents. Act can be divided into the sets Act_{own} of the agent's actions not dealing with other agents and Act_{co} of its communication and cooperation actions. The internal data areas of the agent can belong to three different sets, namely the set Dat_{own} of values of data areas concerning information about the agent *itself*, the set Dat_{sag} of values of data areas concerning *sure* knowledge about other agents, and the set Dat_{usag} of values of data areas containing *unsure* knowledge about the other agents.

The agent architecture we will use for our agents is rather simple. The data areas (Dat_{own}) of an agent holds a set of prototypical *situation-action-pairs*, SAPs, (i.e. $Dat_{own} = 2^{Sit \times Act}$) and its decision function $f_{\mathcal{A}g}$ uses the nearest-neighbor rule to determine the action to the actual situation. More precisely, for all SAPs (s_i,a_i) in the actual value of Dat_{own} the distance of s_i to the actual situation s with respect to a distance measure f_{dist} is computed (in our case f_{dist} computes the Euclidean distance of two number vectors). Then the action of the pair with the smallest distance (or greatest similarity to the current situation) is chosen (i.e. $a = a_j$ with $f_{dist}(s_j,s)$ is minimal). Note that we also can use *enhanced* situation descriptions, in which a situation is not just an element of Sit, but can contain additional data about the agent. In general, this agent architecture allows only for rather reactive agents.

2.2 Evolutionary Learning of SAPs

The evolutionary learning approach used in [3] and [4] focuses on evolving suitable SAP-sets. For all learning agents, a genetic algorithm evolves their set of SAPs, their *strategy*. An individual of the population maintained by the GA consists of several sets of SAPs, one set for each agent. As usual, we use crossover and mutation as genetic operators. The mutation operator just deletes an arbitrary number of SAPs in a strategy and then adds some randomly generated new SAPs. For each agent within an individual, the crossover operator randomly selects SAPs from both parents and puts them together to form the offspring.

The fitness computation for the evolutionary learning of cooperative behavior is based on a restricted simulation of the multi-agent system which measures the agent's behavior. The simulation is restricted because the number of steps that the agents are allowed to do is limited. In all other regards the simulation is identical to the task. The result of a single simulation run is called the behavior B. Starting from situation s_0 for a set of strategies for the learning agents $\mathcal{A}g_1$,...,$\mathcal{A}g_k$ we get $B(\mathcal{A}g_1,...,\mathcal{A}g_k,s_0) = s_0,s_1,...,s_{i-1},s_i,...$. If the task can be solved by the agents represented by the individual, let's say with situation s_i, then the number of steps, i, is the fitness measure. Else, a measure of how near the agents came to doing the task is used. Since we are measuring the behavior of the agent strategies, each situation produced during a run should have some influence on the fitness. Naturally, the measure has to be based on the problem to solve and therefore can only be given for a concrete application. We will do this in the next section. If there are random effects occurring during simulation runs, the fitness measure should take into account several simulation runs, to deal with these random effects. And as usual, the genetic algorithm favors fitter individuals for use by the genetic operators.

3 Pursuit Games and the OLEMAS System

In order to evaluate the evolutionary learning method presented in the last section, we build the OLEMAS system, that uses Pursuit Games as the application domain. Pursuit Games as a testbed for multi-agent systems and the cooperation of agents were first suggested in [1]. Since then, many authors have used variants for testing their approaches. The basic idea of all variants is that one or several hunter agents have to catch one or several prey agents within a world that usually is represented as a collection of connected grids. Time in the world is expressed as a sequence of turns. When used as a testbed for learning agents, usually the hunters (or some of them) are the learning agents, although some authors also looked at learning prey agents.

The following features of a Pursuit Game can be varied in the OLEMAS system (see [3]):

1. **The form of the grid**
   In the basic scenario described in [1] the grid-world has no boundaries and there are no obstacles in it. OLEMAS allows for variants by introducing boundaries (e.g., a $N \times N$ grid) or obstacles (with varying shapes). Obstacles are treated as agents (a kind of innocent bystanders) and they are even allowed to move.
2. **The individual hunter**
   In the basic scenario a hunter agent does not have many features. But we can obtain variants with respect to the following sub-features:
   a) *Shape and size*
      A hunter may occupy one or more grid squares. It may be rectangular, but it can also have other shapes. OLEMAS allows any set of connected points (occupying a grid) for the agent's shape.
   b) *Possible moves and actions*
      Besides moving only in the vertical or horizontal direction (or not moving at all) variants include diagonal moves or rotations, if rotations actually have an effect. In theory, there can also be communication actions but so far we do not have a communication action in OLEMAS.
   c) *Speed*
      Hunters, like all agents, have with each action associated a number of game turns that are needed to perform the action.
   d) *Perception and communication capabilities*
      An aspect that greatly influences the strategy of a hunter (and therefore each solution attempt for the game) are its perception and communication capabilities. (Note that being able to see the other hunters is a kind of visual communication.) The spectrum of this aspect ranges from hunters with no or very limited communication capabilities to hunters utilizing sophisticated communication methods. In OLEMAS, we allow for hunters to either see only the prey agents or all agents.
   e) *Memory capabilities*
      An aspect that can become important if the hunters are able to communicate with each other is the memory of an agent. Memory allows

an agent to remember plans and intentions of other hunters. There may be no memory, a restricted size for the memory, or an arbitrary amount of memory. In OLEMAS, memory can only be introduced in form of extended situations.

3. **The hunting team**
   For most variants of the game more than one hunter (and cooperation between the hunters) are required for the team to succeed. Therefore, the composition of the team of hunters is also an aspect that can be varied.
   a) *The number of hunters*
      For each combination of the other aspects there is a minimal number of hunters needed to win the game. Deploying more hunters may help to win the game, but may also require different, possibly more sophisticated strategies and more effort in developing these strategies.
   b) *The type of the hunters*
      Since there can be different types of hunters, quite different strategies –depending on what kind of hunters form the team– are needed to cooperate in order to win.
4. **The prey**
   The prey is an agent like the hunters. Therefore the same sub-features a) to e) apply with the exception that communication is only necessary if there are several prey agents. But there is an additional sub-feature:
   f) *The strategy of the prey*
      The (escape) strategy of the prey is the main factor determining the difficulty to win the game. Strategies range from simply moving in one direction to random moves to elaborate strategies like maximizing the distance from all hunters, obstacles and boundaries. OLEMAS also allows for using a set of SAPs and the nearest-neighbor rule as the strategy for a prey.
5. **The start situation**
   The start positions of both hunters and prey can also influence both the possibility to win the game and the effort for learning a cooperative strategy for the hunters. If the game is always started from the same positions and no random element is introduced by other aspects, then a winning strategy will always win (and is easier to learn). Otherwise, different start situations will lead to different outcomes.
6. **The game goal**
   Even for the definition of if and when the game is won there are different variants. The main question is to "capture" or to "kill". The prey is captured if it cannot move to another grid anymore (i.e., it is totally surrounded by boundaries, obstacles and hunters). If a hunter occupies the same grid as the prey (at some point in time), the prey is killed. With several prey agents, the number of caught prey agents has to be defined within the game goal. Usually, the goal also includes resource limitations, i.e. the number of turns T_{Max} within which the goal has to be fulfilled.

With so many features to vary, there is an enormous number of game variants and an important goal in [3] was to define situation descriptions for SAPs and a fitness measure that allows a hunter to learn winning strategies for many of them. For a human observer, the situations occurring during a game run are characterized by the positions and orientations (if an agent occupies more than one grid) of all agents on the grid field. The internal representation of a situation makes this more precise by stating the positions of the agents in a predefined order and by stating a position as the coordinates of the so-called centerpoint of each agent (that is a property of the agent). For a situation in a SAP in the strategy of a particular learning agent, we transform the objective game situation into the subjective view of the learning agent by presenting the coordinates of the other agents relative to the learning agent. This also allows us to filter out agents that the learning agent is not allowed to see according to the particular game variant.

As described in the last section, if the fitness *fit* of an individual $\mathcal{I} = \mathcal{A}g_1,...,\mathcal{A}g_k$ is based on evaluating b different runs $r_1,...,r_b$, then we have

$$fit(\mathcal{I}) = \frac{1}{b} \cdot \sum_{i=1}^{b} efit(r_i, \mathcal{I}),$$

The elemental fitness *efit* of a run r_j with the behavior $s_{j0},s_{j1},...,s_{j,T_{Max}}$ is defined as

$$efit(r_j, \mathcal{I}) = \begin{cases} l, & \text{success in } l \leq T_{Max} \text{ steps} \\ \sum_{t=1}^{T_{Max}} success(s_t), & \text{in case of failure,} \end{cases}$$

and the function *success* that measures how near a particular encountered game situation s_t is to a winning situation is approximated in OLEMAS by summing up the Manhattan distances of all hunters from all preys. The Manhattan distance $dist_{Manh}$ between two agents at coordinates (x_1,y_1) and (x_2,y_2) is defined as

$$dist_{Manh}((x_1, y_1), (x_2, y_2)) = |x_1 - x_2| + |y_1 - y_2|$$

We also use the Manhattan distance in the nearest-neighbor rule to measure the similarity of two situations. The Manhattan distances between the coordinates of an agent in both situations is computed for all agents that are visible and these distances are summed up to produce the similarity measure. If the similarity to the current situation is the same for two or more situations in the set of SAPs forming the agent strategy, then the action of the first SAP is taken.

4 Putting More Knowledge into the Fitness Function: Case Studies

As already stated, the fitness function has always been considered as an appropriate place in an evolutionary algorithm to add knowledge that someone has about the problem he or she wants to solve. But many authors also reported on the difficulties of incorporating more knowledge into the fitness function. The knowledge has to be represented rather indirectly to be used to better distinguish good individuals from bad ones. This brings the risk of focusing too narrowly, thus guiding the algorithm to only local optima or to deadends without solutions.

In case of our evolutionary learning, the fitness function does not measure individuals directly, it measures the cooperative behavior that is induced by the individuals, thus making any knowledge representation even more indirect and more prone to divert from the main objective. By defining the elemental fitness *efit* of a run of an individual based on the success evaluation of all encountered situations, we already introduced a more direct way to incorporate knowledge by concentrating on what is good or bad in a particular situation. But there are many aspects of a situation that might be of interest regarding how near to success we are and usually there are different ways to measure a particular aspect.

If we look at the base fitness function defined in the last section, the key aspect is the distance between agents. To measure distance, we used the Manhattan distance, which can be easily computed from the coordinates of the agents. But there are additional, more accurate measures of distance. Will their use improve our learning? Also, agents with shapes and grids with obstacles add an additional dimension to the concept of distance. If I am near to someone else if we ignore things like obstacles, then I can still be far away from the other agent if I cannot fly and have to walk around the obstacle. And, depending on my shape, spaces between obstacles might be too small to pass through or require rotations at the right times. And what about the goal of the pursuit game? If we have to catch several prey agents, should this not be reflected in how we measure being near to a goal-fulfilling situation? In the following, we will present modifications of our base fitness function that reflect these ideas of additional knowledge and we will evaluate the usefulness of these new fitness functions, compared to the base fitness, with pursuit game variants that benefit from using the additional knowledge.

4.1 General Remarks

All of the experiments we present in the following subsections report on changes we made to the function *success*, defined in Section 3, to reflect certain knowledge about game features. Often, these changes are adding a certain evaluation criterion to the base *success*-function.

Whenever we combined several criteria $C_1(s)$,...,$C_m(s)$ for a situation s, we first normalized the individual criteria and then used weighting factors w_1,...,w_m to control the influence of the criteria (although, due to lack of space, we will report on one weight combination only). So, the general combination function *success-comb* we used is defined as follows:

$$\textit{success-comb}(C_1, ..., C_m)(s) = \sum_{i=1}^{m} w_i * \textit{norm}(C_i(s))$$

A criterion is normalized by dividing its value by the maximum possible value for the criterion. However, for many of the criteria we developed, this maximum value is difficult to determine a priori or the values occurring during a run are far from the theoretical maximum, thus requiring quite some fiddling with the criterion weight to get the influence of the criterion right. Therefore we decided to use the criterion value of the start situation as the value for normalization, if the criterion is not limited to only a few possible values.

For all of our experiments, we report on the success rates of the learner and average number of turns required by the successful strategy. Since an evolutionary algorithm uses random decisions, every run of the algorithm is different, so that we have to report a success rate (in percent). Naturally, the average number of turns needed by a strategy is computed for the successful runs only. We also provide the standard deviation we observed among the successful runs with regard to turns needed by the hunters using the learned strategies to win the game. We report on the average number of generations needed to find a winning strategy only if there was a substantial difference between the different fitness functions.

In all of the experiments, the agents can move in all eight directions and, if they have an irregular shape, they can rotate 90 degrees to the left and right, unless otherwise stated. Each action takes one turn to accomplish. All game variants are played on a 30 by 30 grid world and the hunters have 200 turns to win the game. If there is only one prey, then the game goal is to kill it. In all learning runs, we stop when the first strategy winning the game is found.

4.2 Does More Precision Help?

The base function used for function *success* is based on measuring the distance between hunters and preys using the Manhattan distance $dist_{Manh}$. This is not the most primitive way to determine a distance, but also not the most accurate way. To understand how different distance computations influence the results, we tested two other ways, namely $dist_{max}$ that uses the maximum of the coordinate differences in the x- and y-axis:

$$dist_{max}((x_1, y_1), (x_2, y_2)) = \max\{|x_1 - x_2|, |y_1 - y_2|\}$$

and $dist_{Eucl}$ that uses the Euclidean distance between the coordinates of the two agents:

$$dist_{Eucl}((x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

The game scenario we used to come up with the results from Table 1 is as follows: We have a simple scenario with one hunter and one prey, see Variant 1 in Figure 1 for the start situation. The prey tries to avoid the hunter and all borders. The parameters for the GA were a population size of 100 for a maximum of 10 generations with a mutation rate of 15 percent and 20 individuals surviving from generation to generation. The strategies for the hunter could contain at most 20 SAPs. We did 100 learning runs per distance function.

The results in Table 1 are somewhat surprising. While more precision in the distance evaluation allowed for 5 percent more success, producing solutions of approximately the same quality, being more crude in measuring the distance

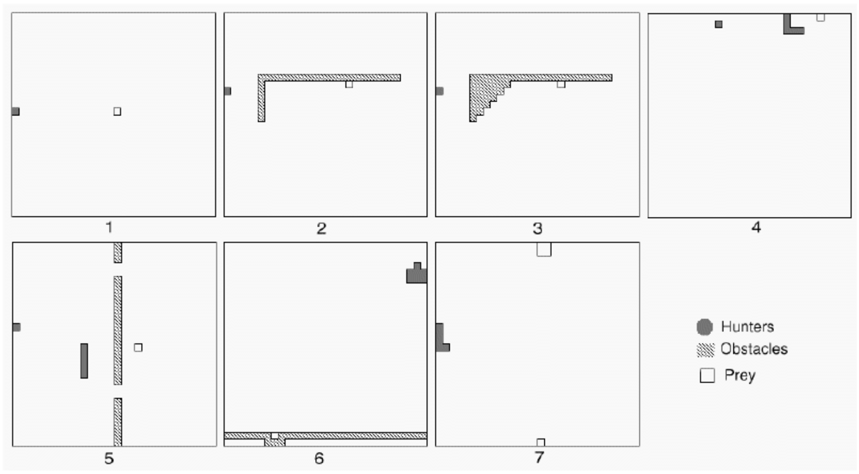


Fig. 1. Start situations for game scenarios

Table 1. Experimental results for different distance measures

Distance function	Success rate	Steps	Std. deviation (Steps)
$dist_{Manh}$	79%	58.6	23.6
$dist_{max}$	90%	56.2	19.2
$dist_{Eucl}$	84%	57.7	24.6

gives us an increase of 11 percent and more stability with regard to the quality of the solution. So, here we have an example where more precision in measuring knowledge about a feature does not really help.

4.3 Not Every Grid-Square Should Count the Same!

For Pursuit Game variants where agents occupy just one grid-square and where there are no obstacles, using any of the distance measures we looked at in the last subsection works well. The picture changes drastically if we have obstacles in the game variant and/or agents have shapes with a lot of blocking potential. Since getting by an obstacle or other agent often means having to go away from the preys, the base fitness of measuring how near the hunters are to the preys often leads the learning agents into a local optimum. Only with a lot of luck, such as a combination of several SAPs that together manage to get completely around the obstacle, can the learning still be successful. In [4], we showed that the problem can be reduced by using on-line learning, but obstacles in the way still is a basic problem in many game variants with regard to learning.

In order to deal with such game variants, the fitness function must account for obstacles and other agents between the learning agent and the prey, making these situations less good than situations where the path to the prey is clear. Obviously, this means that we have to look at the paths from the hunters to the preys and have to measure grid-squares on these paths according to them being occupied or not. For a grid-square gs we define its *terrain value* v_{terr} as

$$v_{terr}(gs) = \begin{cases} H \text{ if } gs \text{ is occupied by a moving agent} \\ O \text{ if } gs \text{ is occupied by an obstacle} \\ E \text{ if } gs \text{ is not occupied} \end{cases}$$

In our experiments, we used $E = 1$, $H = 10$, and $O = 15$.

Then, the terrain distance $dist_{terr}$ for a sequence gs_1,...,gs_k of grid-squares is defined as

$$dist_{terr}(gs_1, ..., gs_k) = \sum_{i=1}^{k} v_{terr}(gs_i).$$

Naturally, the question remains how to get the sequences of grid-squares between two agents, since there are several possibilities. In our experiments, we tested three possibilities related to the Manhattan distance:

$dist_{terr,x}((x_1,y_1),(x_2,y_2))$: starts at (x_1,y_1) moves along the x-axis to (x_2,y_1) and then moves along the y-axis to (x_2,y_2)

$dist_{terr,y}((x_1,y_1),(x_2,y_2))$: starts at (x_1,y_1) moves along the y-axis to (x_1,y_2) and then moves along the x-axis to (x_2,y_2)

$dist_{terr,xy}((x_1,y_1),(x_2,y_2))$: $dist_{terr,x}((x_1,y_1),(x_2,y_2))$ + $dist_{terr,y}((x_1,y_1),(x_2,y_2))$

The reason for the different ways of generating the grid sequence becomes clear if we take a look at the start situations depicted in Figure 1 for variants 2 to 5. If we compare variants 2 and 3, we can see that the distance values differ quite a bit already in the start situations depending on whether we go first right or first up. The game variants 2 to 5 are as follows. In variants 2 and 3 we want to focus on getting the hunter around the obstacle. Therefore the prey does not move at all. Note that the hunter will not be successful within the given number of turns if it tries to pass above the obstacle, since it has to go a very long way (and away from the prey for quite a number of steps). For both variants the GA used 100 individuals for a maximum of 10 generations. We used a mutation rate of 25 percent and a strategy had at most 20 SAPs. The results in Table 2 are based on 100 learning runs for each fitness function.

In variant 4, we have two hunters that both learn. The L-shaped hunter is not allowed to rotate. The prey evades the nearest hunter and thus ends up in the upper right corner of the world. The problem of the hunters is to figure out that the small hunter has to pass the L-shaped one, because otherwise the L-shaped hunter ends up “protecting” the prey in the corner. This is not so easy to learn and therefore we used 200 individuals per generation and allowed for a maximum of 30 generations. Also, we used a mutation rate of 30 percent and increased the

Table 2. Experimental results with terrain fitness

Terrain fitness		Var. 2	Var. 3	Var. 4	Var. 5
$dist_{Manh}$	Success rate	65%	55%	77%	10%
	Steps	29.2	28.7	34.6	56.7
	Std. deviation (Steps)	8.3	9.1	21.7	16.7
$dist_{terr,x}$	Success rate	86%	82%	100%	37%
	Steps	29.1	29.8	39.0	46.8
	Std. deviation (Steps)	7.9	7.3	26.9	15.8
$dist_{terr,y}$	Success rate	84%	82%	100%	27%
	Steps	29.2	27.4	28.6	41.8
	Std. deviation (Steps)	9.6	9.5	13.6	9.3
$dist_{terr,xy}$	Success rate	88%	92%	100%	37%
	Steps	30.0	29.0	30. 6	54.9
	Std. deviation (Steps)	9.6	9.5	17.0	19.2

maximum number of SAPs per individual to 30. As a consequence, the runtime of the learner went up and we report only on the results of 30 learning runs per function.

Variant 5 uses the same parameters for the GA as variant 4. Also the prey strategy is the same. The bigger hunter is not learning, it just moves towards the prey. It is not allowed to rotate. The small hunter is the learning hunter and the problem it has to deal with is that the larger hunter blocks the upper passage through the obstacle while moving towards the prey. Therefore it has to learn to pass through the lower passage. So, variant 5 combines the difficulties of the other 3 variants.

As Table 2 shows, using all 3 terrain distance measures improves the success rate of the learning drastically. The combination of the two ways to generate paths, $dist_{terr,xy}$, always has the highest success. For variants 2 and 3, the average number of steps and its standard deviation are approximately the same. For variant 4, using $dist_{terr,x}$ results in worse strategies than the default measure, while $dist_{terr,y}$ finds consistently shorter solutions. But the combined terrain fitness is only a little bit worse than that. By using $dist_{terr,y}$, the cooperative strategy of the two hunters that lets the bigger hunter move down to let the smaller one pass is more encouraged than by $dist_{terr,x}$, that requires the big hunter to move down more before the advantage becomes obvious to the fitness measure. Even the standard distance measure can take better advantage of that. But in our experiments with variant 4 we also observed that for all terrain measures the number of generations necessary to find a solution was less than half of the number required by the default measure, which shows that the awareness of the blocking potential also can improve the learning time. For variant 5, the success rate of $dist_{terr,y}$ is 10 percent less than the other terrain measures, but those strategies that were found are consistently better than the ones generated by all other measures.

All in all, using a distance measure that distinguishes terrain features to indicate blocking potential increases the chance for success of the learner. The path we use to calculate the terrain value has some influence on both success rate and the quality of the found solutions. If we are willing to combine the measures for different paths, the problem of influence of the path can be overcome by spending more computing time.

4.4 If You Have a Shape Then Orientation Matters!

Another problem that comes up when we have agents with irregular shapes is that usually we require these agents to orient themselves into a certain direction (by rotating), so that their shape fits right into what is required for winning the game. If a hunter tries to rotate too late in the game, it might not be possible anymore due to obstacles blocking a rotation. Then we can fall into a local optimum without success. One could argue that making the fitness function aware about what the right orientation for a hunter is, is already giving away part of the solution to win the game and this is definitely true. But obstacles might require rotations to be passed by and, as our experiments show, it is not so easy to express the "right" orientation via the fitness function. In addition, our experiments also highlight that choosing the centerpoint of an agent and its initial orientation can have rather dramatic effects on the performance, regardless of what we do with the fitness function.

In OLEMAS, when defining an agent, we define its shape, indicate within the shape the centerpoint of the agent and we define a preferred *front* of the agent, which is the side of the agent initially given as the top of its shape. For indicating within a fitness function, resp. the *success*-function within the fitness function, how a hunter agent h is oriented with regards to a prey p, we define the orientation criterion C_{orient} as follows. We partition the whole grid world into 4 sections that are defined based on the centerpoint of the hunter by two diagonal lines passing through this centerpoint. The front section is the section of the world that is faced by the front of the agent, the sections to the right and left are obviously the right and left section and the final section is called the rear section. Then we have for a situation s:

$$C_{orient}(h,p,s)\begin{cases}0 \text{ if centerpoint of p is in front section}\\ 1 \text{ if centerpoint of p is in right or left section}\\ 2 \text{ if centerpoint of p is in rear section}\end{cases}$$

This criterion is normalized by dividing by 2 and we combine it with the criterion defined by using $dist_{Manh}$.

Variant 6 in Figure 1 shows the starting situation for the Pursuit Game variant we have chosen to illustrate the influence of the criterion C_{orient}. As in variants 2 and 3, the prey does not move, thus forcing the hunter to rotate to achieve the right orientation, so that its "nose" can kill the otherwise protected prey. The GA parameter settings were similar to variants 2 and 3, we only changed the maximum number of generations to 30. Since we measure the distance between two agents from centerpoint to centerpoint, already the placement

Table 3. Experimental results with orientation

Fitness		Var. 6a)	Var. 6b)	Var. 6c)
$dist_{Manh}$	Success rate	36%	73%	75%
	Steps	27.3	25.6	25.4
	Std. deviation (Steps)	4.1	2.3	4.0
	Nr. of generations	15.5	13.9	11.7
C_{orient}, with $w_{orient} = 1$	Success rate	68%	92%	77%
	Steps	30.8	29.0	26.1
	Std. deviation (Steps)	9.8	8.6	4.8
	Nr. of generations	15.7	11.1	12.2
C_{orient}, with $w_{orient} = 10$	Success rate	55%	79%	53%
	Steps	42.2	29.8	37.6
	Std. deviation (Steps)	17.6	9.5	22.4
	Nr. of generations	22.2	16.8	18.4
C_{orient}, with $w_{orient} = 100$	Success rate	45%	65%	48%
	Steps	39.2	30.9	35.8
	Std. deviation (Steps)	13.2	11.6	17.7
	Nr. of generations	20.1	17.9	19.5

of this centerpoint should have some influence on the outcome and therefore we tried out 3 different placements, namely

a) in the hunter's nose
b) in the center of the hunter's body and
c) on the hunter's back directly opposite to the nose.

As Table 3 shows, the placement of the centerpoint has already quite some effect on the success rate and the learning time for the default fitness alone. The wrong placement results in half of the success rate and approximately 4 more generations needed to find the successful strategies. Adding the orientation criteria with equal weight as the distance criterion improves the success rate for each of the centerpoint placements, but helps especially in the case of a bad placement. While the learning times are comparable, the additional success is paid for by worse solution quality that varies much more than without the orientation criterion, except for the case c) where distance alone is already quite good. Increasing the influence of the orientation criterion does not improve the help by orientation, in fact it results in longer learning times and worse solution quality. Especially if the centerpoint placement is already good for the standard fitness, an increase of the influence of C_{orient} results in a detoriation of the learning performance.

All in all, placement of the centerpoint has shown to be potentially crucial for the success of learning. By including orientation into the fitness, we can make the influence of centerpoint placement less crucial and we can improve the success rate in general.

Table 4. Experimental results with prey clustering

	Success rate	Steps	Std. deviation (Steps)
C_{prey}, with $w_{prey} = 0$	14%	43.9	14.9
C_{prey}, with $w_{prey} = -0.1$	20%	55.8	20.8
C_{prey}, with $w_{prey} = -1$	29%	48.5	13.3
C_{prey}, with $w_{prey} = -10$	41%	51.9	16.7

4.5 Complex Catches Require More Complex Evaluations!

With certain Pursuit Game variants it is possible to mimic some problem aspects of many other application areas for learning agent behavior. *Herding* is one of these application areas (as suggested in [11]). As a Pursuit Game variant, bringing prey agents together into a herd can be achieved by having as a game goal that the hunter(s) kill all the prey, which means that the hunter(s) must, in the same turn, occupy for each prey at least one of the grids occupied by the prey agents. If we have only one hunter (or less hunters than prey), then the knowledge we should convey to the learner is that it is good to have prey agents stay together.

Since the hunters have to "overlap" prey agents to achieve the game goal, we found a simple distance measure for preys, similar to what we used for variant 1, not sufficient. Instead we assigned to each prey a so-called *control radius* that extends around it from the farthest grid occupied by the prey to all sides. For two prey, p_1 and p_2, we define their x-overlap and y-overlap in a situation s as follows:

x-overlap(p_1,p_2,s) = number of grid squares along the x-axis in which the control radii of p_1 and p_2 overlap,
y-overlap(p_1,p_2,s) = number of grid squares along the y-axis in which the control radii of p_1 and p_2 overlap.

And then we have as the overlap:

overlap(p_1,p_2,s) = x-overlap(p_1,p_2,s) × y-overlap(p_1,p_2,s)

With this we can define the prey criterion $C_{prey}(s)$ in a situation s by summing up the overlap of each pair of prey agents. This is normalized by dividing by $2 \times radius + 1$ with a *radius* of 15 used in our experiments. Since the more overlap the better, we use as w_{prey} negative values.

To evaluate C_{prey}, we used a game scenario with start situation 7 in Figure 1. The big prey evades just the hunter, while the smaller one in addition tries to stay away from the borders. The GA parameters are similar to variant 6, except that we allow for 40 SAPs in an individual.

As Table 4 shows, the additional C_{prey} criterion increases the success rate substantially, it nearly triples it, if we use $w_{prey} = -10$. This comes with the price of finding longer solutions, but this is definitely acceptable. So, taking knowledge about the game goal into account again results in an improvement.

5 Conclusion and Future Work

We presented several case studies showing how knowledge about application features can be integrated into an evolutionary learner for cooperative behavior of agents. Our experiments showed that awareness of a certain feature substantially increases the success of the learner (assuming limited resources for the learning), while increased precision of the knowledge already used is not guaranteed to improve the learning.

Obviously, there are many more features of groups of Pursuit Game scenarios that can be useful for an evolutionary learner and that we want to look into in the future. Our interest will be in finding features that are not typical for just a few game variants but useful in many of them, as has been the terrain fitness. Naturally, we also are interested in identifying additional ways to include knowledge into learning behavior. And it will be interesting to see if there are other "hidden" features, like the placement of the centerpoint, that influence learning performance.

References

1. M. Benda; V. Jagannathan and R. Dodhiawalla. An Optimal Cooperation of Knowledge Sources, Technical Report BCS-G201e-28, Boeing AI Center, 1985.
2. J. Denzinger and S. Ennis. Being the new guy in an experienced team – enhancing training on the job, Proc. AAMAS-02, ACM Press, 2002, pp. 1246–1253.
3. J. Denzinger and M. Fuchs. Experiments in Learning Prototypical Situations for Variants of the Pursuit Game, Proc. ICMAS'96, AAAI Press, 1996, pp. 48–55.
4. J. Denzinger and M. Kordt. Evolutionary On-line Learning of Cooperative Behavior with Situation-Action-Pairs, Proc. ICMAS'00, IEEE Press, 2000, pp. 103–110.
5. J. Denzinger and M. Kordt. On the influence of learning time on evolutionary online learning of cooperative behavior, Proc. GECCO-2001, Morgan Kaufmann, 2001, pp. 837–844.
6. G. Eiben and J. van Hemert. SAW-ing EAs: Adapting the Fitness Function for Solving Constrained Problems, in: Corne, Dorigo, Glover (eds.): New Ideas in Optimization, McGraw-Hill, 1999, pp. 389–402.
7. F. Gomez and R. Miikkulainen. Incremental Evolution of Complex General Behavior, Adaptive Behavior 5, 1997, pp. 317–342.
8. T. Haynes, R. Wainwright, S. Sen and D. Schoenefeld. Strongly typed genetic programming in evolving cooperation strategies, Proc. 6th GA, Morgan Kaufmann, 1995, pp. 271–278.
9. K.-C. Jim and C.L. Giles. Talking helps: Evolving communicating agents for the predator-prey pursuit problem, Artificial Life 6(3), 2000, pp. 237–254.
10. J.D. Schaffer, D. Whitley and L.J. Eshelman. Combinations of genetic algorithms and neural networks: A survey of the state of the art, Proc. COGANN-92, IEEE, 1992, pp. 1–37.
11. A. Schultz, J. Grefenstette and W. Adams. Robo-shepherd: Learning complex robotic behaviors, Robotics and Manufacturing: Recent Trends in Research and Application, Volume 6, ASME Press, 1996, pp. 763–768.
12. P. Stone. Layered Learning in Multi-Agent Systems: A Winning Approach to Robotic Soccer, MIT Press, 2000.

Towards Efficient Training on Large Datasets for Genetic Programming

Robert Curry and Malcolm Heywood

Dalhousie University, Faculty of Computer Science, 6050 University Avenue, Halifax, Nova Scotia, Canada, B3H 1W5
{rcurry, mheywood}@cs.dal.ca

Abstract. Genetic programming (GP) has the potential to provide unique solutions to a wide range of supervised learning problems. The technique, however, does suffer from a widely acknowledged computational overhead. As a consequence applications of GP are often confined to datasets consisting of hundreds of training exemplars as opposed to tens of thousands of exemplars, thus limiting the widespread applicability of the approach. In this work we propose and thoroughly investigate a data sub-sampling algorithm – hierarchical dynamic subset selection – that filters the initial training dataset in parallel with the learning process. The motivation being to focus the GP training on the most difficult or least recently visited exemplars. To do so, we build on the dynamic sub-set selection algorithm of Gathercole and extend it into a hierarchy of subset selections, thus matching the concept of a memory hierarchy supported in modern computers. Such an approach provides for the training of GP solutions to data sets with hundreds of thousands of exemplars in tens of minutes whilst matching the classification accuracies of more classical approaches.

1 Introduction

The interest of this work lies in providing a framework for efficiently training genetic programming (GP) on large datasets. There are at least two aspects to this problem: the cost of fitness evaluation and the overhead in managing datasets that do not reside within RAM alone. The computational overhead associated with the inner loop of GP fitness evaluation has been widely recognized. The traditional approach for addressing this problem has been hardware based. Examples include Beowulf clusters [1], parallel computers [2] and FPGA solutions [3]. In this work we propose to address the problem through the following two observations. Firstly, within the context of supervised learning, the significance of data sampling algorithms have been widely acknowledged albeit with the motivation to improve error performance, e.g. boosting and bagging [4, 5]. Secondly, memory hierarchies are widely used in CPU architectures, where such hierarchies are based on the concept of temporal and spatial locality [6]. The motivation used here, however, is that any learning algorithm need only see a subset of the total dataset, where the sampling process used to identify such a subset of exemplars should also be sympathetic to the memory hierarchy of the computing platform.

To address these issues the method of Dynamic Subset Selection [7] was revisited and extended to a hierarchy of subset selections. Such a scheme was applied to the 10% KDD-99 benchmark, a dataset consisting of approximately half a million patterns [8]. The dataset was first partitioned into blocks that were sufficiently small to reside within RAM alone. Blocks were then chosen from this partition based on Random Subset Selection (RSS). This forms the level 1 of the selection hierarchy. At level 2, the method of Dynamic Subset Selection (DSS) was used to stochastically select patterns from the block identified at the first level. Several rounds of DSS are performed per level 1 block, with exemplars identified on the basis of their difficulty and age. This hierarchy shall be referred to as a RSS-DSS hierarchy.

In this work we concentrate on the parameterization of the algorithm and extending the evaluation to another dataset (the work of Song *et al.*, concentrated on application issues associated with the application to an intrusion detection problem [8]). Furthermore, an alternative hierarchy is introduced in which the level 1 block selection was also chosen using DSS. This new hierarchy shall be referred to as the DSS-DSS hierarchy and is shown to improve the error properties of the ensuing solution.

The remainder of the paper consists of the methodology for hierarchical DSS, the ensuing Results and Discussion, Sections 2, 3 and 4 respectively.

2 Methodology

As indicated above, our principle interest lies in the investigation of an exemplar subsampling algorithm, which filters the dataset in proportion to the ‘age’ and ‘difficulty’ of exemplars as viewed by the learning algorithm, whilst also incorporating the concept of a memory hierarchy. Such a scheme should significantly decrease the time to complete the inner loop of GP, without impacting on the error performance. The algorithm is actually independent of the supervised learning algorithm, but in this case is motivated by the plight of GP in which the inner loop is iterated over a population of candidate solutions. Section 2.1 summarizes the form of GP utilized later (any generic form of GP will suffice), whereas the methodology for hierarchical dynamic subset selection is detailed in Sections 2.2 and 2.3.

2.1 Genetic Programming

In the case of this work a form of Linearly-structured GP (L-GP) is employed [9-12]. That is to say, rather than expressing individuals using the tree like structure popularized by the work of Koza [13], individuals are expressed as a linear list of instructions [9]. Execution of an individual therefore mimics the process of program execution normally associated with a simple register machine. That is, instructions are defined in terms of an opcode and operand (synonymous with function and terminal sets respectively) that modify the contents of internal registers {R[0],…,R[*k*]}, memory and program counter [9]. Output of the program is taken from the best register upon completion of program execution (or some appropriate halting criterion [11]), where the best register is the register of the best performing individual that generates the greatest number of hits. Moreover, in an attempt to make the action of the crossover operator less destructive, the Page-based formulation of L-GP is employed [12]. In this case, an individual is described in terms of a number of *pages*, where each page has the *same* number of *instructions*. Crossover is limited to the exchange of *single* pages between two parents, and appears to result in concise solutions across a range of benchmark regression and classification problems. Moreover, a mechanism for dynamically changing page size was introduced, thus avoiding problems associated with the *a priori* selection of a specific number of instructions per page at initialization. Mutation operators take two forms. In the first case the 'mutation' operator selects an instruction for modification with uniform probability and performs an Ex-OR with a second instruction, also created with uniform probability. If the ensuing instruction represents a legal instruction the new instruction is accepted, otherwise the process is repeated. The second mutation operator 'swap' is designed to provide sequence modification. To do so, two instructions are selected within the same individual with uniform probability and their positions exchanged.

2.2 RSS-DSS Hierarchy

The basic formulation for the hierarchical sampling of training exemplars divides the problem into three levels. Level 0 divides the training set into a sequence of equal blocks. Blocks reside in memory and are chosen stochastically, Level 1. Level 2 samples the exemplars of the selected block using a stochastic sampling algorithm biased towards the more difficult or older patterns, or Dynamic Subset Selection (DSS) [7]. Program 1 outlines the general relationship between learning algorithm (GP in this case) and hierarchical sampling algorithm for the case of RSS block selection at level 1 and DSS exemplar selection at level 2:

```
Program 1 - Genetic Programming with RSS-DSS Hierarchy
{
(1)  divide dataset into blocks (level 0)
(2)  initialize training system and population
(3)  while (RSStermination == FALSE)
     {
(4)    conduct Block Selection (level 1)
(5)    while (DSStermination == FALSE)
       {
(6)      conduct Subset Selection (level 2)
(7)      while (TournamentEnd == FALSE)
         {
(8)        conduct tournament selection
(9)        train tournament individuals on Subset
(10)       update connection difficulty
(11)       apply genetic operators
         }
       }
(12)   update #Subset selected at next block b instance
     }
(13) run best individual on entire dataset
(14) run best individual on test dataset
(15) record results
(16) remove introns and translate
}
```

Basic design decisions now need to identify: how a block is identified (level 1), how a subset is selected (level 2) where GP individuals only iterate over the contents of a subset, and how many subsets are selected per block, i.e. the source of the computational speedup. Two basic algorithms are proposed, RSS-DSS (following) and DSS-DSS (§2.3).

Level 1 – Block based Random Subset Selection (RSS). At level 0 the datasets are partitioned into 'blocks' (Program 1-1), all the blocks exist on the hard disk. A block is then randomly selected with uniform probability (Program 1-4) – or Random Subset Selection (RSS) – and read into RAM, level 1 of the hierarchical Subset Selection algorithm. Following selection of block '*b*' a history of training pressure on the block is used to determine the number of iterations performed at level 2 of the RSS-DSS hierarchy – the DSS subset. This is defined in proportion to the error rate over the previous instance of block '*b*' for the 'best' individual over a level 2 subset from block '*b*' (Program 1-12), $E_b(i\text{-}1)$. Thus, the number of DSS subset iterations, *I*, on block, *b*, at the current instance, *i*, is

$$I_b(i) = I_{(\max)} \times E_b(i-1)\ . \tag{1}$$

Where $I_{(\max)}$ is the maximum number of subsets that can be selected on a block; and $E_b(i-1)$ is the error rate (number of block misclassifications) of the best-case subset individual from the previous instance, *i*, of block *b*. Hence, $E_b(i)=1 - [hits_b(i)/\ \#patterns(b)]$, where $hits_b(i)$ is the hit count over block *b* for the best-case individual identified over the last level 2 tournament at iteration *i* of block *b*; and *#patterns*(*b*) is the total number of feature vectors in block *b*. Naturally denoting the 'best case' individual relative to those which took part in level 2 competitions has the potential to miss better performing individuals which might reside in the population. However, this would also reintroduce a significant computational cost of the inner loop. Specifically, an array is kept to record hits for each individual in the population. After each tournament the hits of the parents are accumulated while the hits of the newly created individuals are set to zero. The best case individual over block *b* is therefore the individual with the maximum number of hits accumulated.

Level 2 – Dynamic Subset Selection (DSS). A simplified version of DSS is employed in the second level of the selection hierarchy [7]. Once a block has been selected using the above RSS process, patterns within the block are associated with an age and difficulty. The age is the number of DSS selections since the pattern last appeared in a DSS subset. The difficulty is the number of GP individuals that misclassified the pattern the last time it appeared in the DSS subset. Following selection of a block at level 1, all ages are set to one and all difficulties are set to a worst-case difficulty (i.e. no persistence of pattern difficulties or ages beyond a block selection). Patterns appear in the DSS subset stochastically, with a fixed chance of being selected by age or difficulty (%difficulty = 100 – %age). Thus two roulette wheels exist per block, one is used to control the selection of patterns with respect to age and the other difficulty, the roulette wheels being selected in proportion to the corresponding probability for age and difficulty. This process is repeated until the DSS subset is full (Program 1-6), the age and difficulty of selected patterns being reset to the initial values. Patterns that were not selected from the block have their age incremented by one whereas their difficulties remain the same. Currently DSS uses a subset size of 50 patterns, with the objective of reducing the number of computations associated with a particular fitness evaluation. Each DSS subset is kept for six steady state tournaments (4 individuals taking part per tournament) before reselection takes place with up to $I_{(b)}(i)$ selections per block (Program 1-5) – equation (1).

The use of the fixed probability of 70% for difficulty ensures that greater emphasis is given to connections that resist classification, while the 30% for age ensures that easier patterns are also visited in an attempt to prevent over-learning.

2.3 DSS-DSS Hierarchy

The RSS-DSS algorithm makes the implicit assumption that all blocks are equally difficult. The following DSS-DSS algorithm relaxes this assumption. To do so, a block difficulty and age is introduced and used to bias the stochastic selection of blocks in proportion to their relative age and difficulty using roulette wheel selection. Thus, the probability of selecting block *i* is,

$$\text{Block}(i)_{\text{weight}} = \frac{\%\text{diff} \times \text{Block}_{\text{diff}}(i)}{\sum_j (\text{Block}_{\text{diff}}(j))} + \frac{\%\text{age} \times \text{Block}_{\text{age}}(i)}{\sum_j (\text{Block}_{\text{age}}(j))}$$

$$P(\text{block}(i)) = \frac{\text{Block}(i)_{\text{weight}}}{\sum_j (\text{Block}\ (j)_{\text{weight}})} \tag{2}$$

Where %diff is the fixed difficulty weighting (70%) and %age the age weighting (100 - %diff); $\text{Block}_{\text{diff}}(i)$ and $\text{Block}_{\text{age}}(i)$ are the respective block difficulty and age for block *i*; and *j* indexes all blocks.

At initialization each block has an age of one and worst-case difficulty. Therefore, block selection will initially be uniform. The age of a block is the number of RSS block selections since last selected. The difficulty of a block takes the form of a weighted running average, thus persistent across block selections, or,

$$\text{Block}(i, 0) = \text{Block}\ (i - 1);$$
$$\text{Block}(i, \text{j}) = \alpha\, pattern_{\text{diff}}(j) + (1 - \alpha)\ \text{Block}\ (i, \text{j} - 1);\ \forall \text{j} \in \{1, \ldots, P_{\text{subset}}\}$$
$$\text{Block}(i) = \text{Block}(i, P_{\text{subset}})$$

Where $pattern_{\text{diff}}(j)$ is the difficulty of pattern *j*; j indexes all patterns in the subset of P_{subset} patterns; and, α is a constant ($0 < \alpha < 1$), set to 0.1 in this case. The difficulty of a block will be updated before each new level 2-subset selection (step Program 1-11 and before returning to Program 1-5). Since α is small, each of the patterns in subset have the ability to influence the overall block difficulty by a small amount. If the connection difficulty is high, the overall block difficulty will increase, whereas if the connection difficulty is small, the overall block difficulty will decrease. Level 2 of this hierarchy uses the same DSS process as the RSS-DSS hierarchy.

3 Results

As indicated in the introduction, the principle interest of this work is in establishing a framework for applying Genetic Programming to large datasets. To this end experiments are reported using two large datasets: the KDD-99 Intrusion Detection dataset, taken from the 5th ACM SIGKDD Knowledge Discovery and Data Mining Competition (1999) [14]; and the Adult dataset, taken from the UCI Machine Learning Repository [15]. The total number of patterns in the training and test sets of each dataset is listed in Table 1.

Table 1. DataSets: Sizes and Distributions

	KDD-99		Adult	
Connection	Training (10%KDD)	Test (Corrected Test)	Training	Test
Class 0	97,249	60,577	7,508	3,700
Class 1	396,744	250,424	22,654	11,360
Total	493,993	311,001	30,162	15,060

For KDD-99 two partitions are used, 10% KDD for training and Corrected Test for test, as per the original competition [14]. Each pattern is described in terms of 41 features, comprising of 9 basic features and 32 derived features describing temporal and content information. Here we only use the first 8 basic features, but express these in terms of a shift register with 8 taps taken at intervals of 8 sequential patterns. Such a scheme requires that the learning algorithm also identify the useful temporal properties, rather than relying on the *a priori* selected features (see [16]). Each entry of the KDD dataset represents a connection, labeled in terms of one of five categories: Normal, Denial of Service (DoS), Probe, User to Root (U2R) and Remote to Local (R2L). In this work we are only interested in distinguishing between Normal and any of the four attack categories. Moreover, 79% of the training data represent instances of DoS, 20% normal and the remainder Probe, U2R and R2L, Table 2. Thus, as well as representing a large training set it is unbalanced, introducing the possibility for degenerate solutions i.e. a detector that labels every pattern as attack. The test data on the other hand increases the contribution of the smallest attack category, R2L, to 5% of the dataset and introduces 14 attack types unseen during training, Table 2.

Table 2. Distribution of Attacks

Data Type	Training	Test
Normal	19.69%	19.48%
Probe	0.83%	1.34%
DOS	79.24%	73.90%
U2R	0.01%	0.07%
R2L	0.23%	5.2%

The Adult dataset is significantly smaller than KDD (30 thousand as opposed to half a million patterns), Table 1, but is introduced to verify that the proposed algorithms are not making undue use of duplicate properties that might exist in KDD. Specifically, the DoS class making up nearly 80% of the training data might well demonstrate self-similar properties (a natural characteristic of a DoS attack). Thus, resulting in the sampling algorithm ignoring these exemplars with little or no penalty once some classification capability on DoS has been identified. The adult dataset represents a prediction problem taken from the 1994 Census database. The learning task is to predict whether a person's income exceeds $50,000 per year based on 14 census data features. Each pattern represents a person and is made up of 14 personal and demographic characteristics plus a label. All patterns with missing features were removed from both the training and test data. The data distribution for the Adult dataset is approximately 25% class 0 and 75% class 1 for both training and test sets.

All the following experiments are based on 40 GP runs using Dynamic page-based Linear-GP [12]. Runs differ in their choice of random seeds used for initializing the population, all other parameters remaining unchanged. Table 3 lists the common parameter settings for all runs.

In addition to validating the use of the proposed hierarchical process for sampling patterns, experiments with different block sizes were made under the Adult dataset, Table 4. Note for each dataset the final block does not contain the full block size but the remaining number of patterns.

Finally, a relatively small study is made using a standard C code implementation of Koza's Tree structured GP on the Adult dataset (KDD was too large to run GP on directly without the DSS algorithm). The principle objective being to demonstrate that the proposed DSS-DSS algorithm does not detract from the classification accuracy of 'regular' GP.

Instruction Set. The GP instructions employ a 2-address format in which provision is made for: up to 16 internal registers, up to 64 inputs (Terminal Set), 5 opcodes (Functional Set) – the fifth is retained for a reserved word denoting end of program – and an 8-bit integer field representing constants (0-255) [12]. Two mode bits toggle between one of three instruction types: opcode with internal register reference; opcode with reference to input; target register with integer constant. Extension to include further inputs or internal register merely increases the size of the associated instruction field.

Training was performed on a dual G4 1.33 GHz Mac Server with 2 GB RAM. In all 6 experiments were run. Two experiments on the 10% KDD-99 dataset: the first using a hierarchy of RSS-DSS; and the second using the hierarchy of DSS-DSS. Four experiments were conducted on the Adult Dataset using the two hierarchies as well as the two different block sizes (1000 and 5000).

For each GP run in an experiment the best individual is recorded and simplified by removal of structural introns [17]. The 'best' individual is identified post training by taking the maximum of the hits accumulated by each individual on the entire training dataset. The performance of each best individual is expressed in terms of time, program length (before and after simplification), detection rate (DR) and false positive

Table 3. Parameter Settings for Dynamic Page-based Linear GP

Parameter	Setting
Population size	125
Maximum # of pages	32
Page size	8 instructions
Maximum working page size	8 instructions
Crossover probability	0.9
Mutation probability	0.5
Swap probability	0.9
Tournament size	4
Number of Registers	8
Instruction type 1 probability	0.5/5.5
Instruction type 2 probability	4/5
Instruction type 3 probability	1/5
Function set	{+,-,*,/}
Terminal set	{0, …, 255} ∪ {pattern features}
Level 2 subset size	50
RSS iterations	1000
Max DSS iterations (6 tourn./iteration)	100
Wrapper function	0 if output ≤ 0, otherwise 1
Cost function	Increment by 1/(# in class) for each mis-classification

Table 4. Dataset block sizes and number of blocks

Dataset	10% KDD-99	Adult	
Block size	5000	1000	5000
# of blocks	99	31	7

Rates (FPR) on Corrected KDD Test set. Detection rates and false positive rates are estimated as follows,

$$\text{Detection Rate} = 1 - (\ \text{\# of False Negatives} / \text{Total \# of Attacks}\ )\ . \tag{5}$$

$$\text{False Positive Rate} = (\ \text{\# of False Positives} / \text{Total \# of Normals}\ )\ . \tag{6}$$

3.1 10% KDD-99 Dataset

Results are initially summarized in terms of first, second (median) and third quartiles for time, detection rate, false positive rate, and solution complexity before and after intron removal. In addition comparisons are made against reported figures for both training characteristics and quality of the resulting solutions.

Table 5 indicates that in general the RSS-DSS hierarchy typically takes less time to train than the DSS-DSS hierarchy on the 10% KDD-99 dataset. This appears to be a natural reflection of the overhead in conducting an additional roulette wheel based calculation for block selection in DSS-DSS, as opposed to the uniform selection in the

Table 5. Performance of RSS-DSS and DSS-DSS on 10% KDD-99 Dataset

Algorithm	RSS-DSS	DSS-DSS
	Run Time (minutes)	
1st Quartile	9.17	9.58
Median	10.30	12.28
3rd Quartile	13.45	13.95
	Detection Rate	
1st Quartile	0.8944	0.8917
Median	0.8981	0.8972
3rd Quartile	0.9131	0.9036
	False Positive Rate	
1st Quartile	0.0278	0.0196
Median	0.0493	0.0298
3rd Quartile	0.0794	0.0473
	Solution Complexity – Before Intron Removal	
1st Quartile	111	105
Median	135	167
3rd Quartile	215	199
	Solution Complexity – After Intron Removal	
1st Quartile	19	28.25
Median	51	67.5
3rd Quartile	78	81.25

RSS-DSS algorithm. However, both algorithms are two orders of magnitude better than previously reported research in which GP was applied directly to the entire 10%KDD dataset [18]. Such a direct application required 48 hours to complete a single trial (Pentium III, 800MHz), whereas here each trial takes less than 15 minutes; 40 trials therefore completing in 10 hours.

Detection and False Positive (FP) Rates of the 'best' individual across the 40 runs, Table 5, indicates that uniform selection of blocks results in a significantly wider spread in both Detection and FP rates relative to that from DSS-DSS. Moreover, the median FP rate of DSS-DSS is better than that for RSS-DSS. This property also appears to have carried over to the complexity of solutions, Table 5, with DSS-DSS returning longer solutions (i.e. more instructions) both before and after simplification. Moreover, the trend continues with respect to classification count of each category of attack, Figures 1 - 4. It is also apparent that DSS-DSS has emphasized solutions to the DoS category, where this represents 80% of the training/ test data.

In order to qualify whether the hierarchical subset selection algorithm negatively impacts classifier quality, comparison is made against the results from the original KDD-99 competition winners, Table 6. Both the winning entries were based on decision trees, also taking roughly 24 hours to complete training. The resulting solutions were complex (500 C5 decision trees in the case of the winning entry) and trained using a boosting algorithm on different partitions of the original 10%KDD dataset.

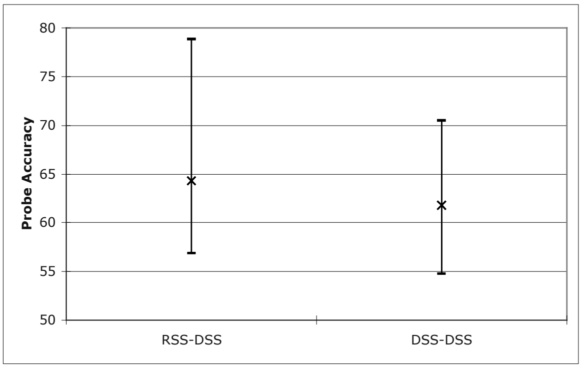


Fig. 1. Probe Accuracy.

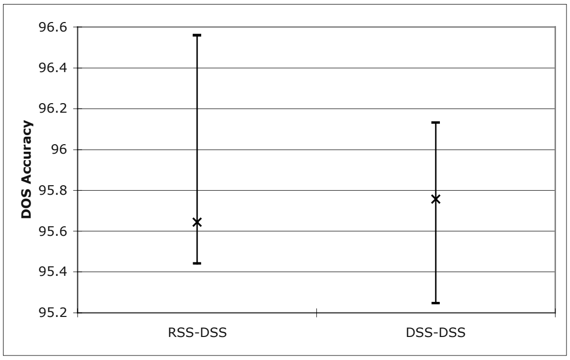


Fig. 2. Denial Of Service Accuracy.

Moreover, independent detectors were developed for each class over all 41 features, making a direct comparison difficult. It is apparent however, that the GP solutions are competitive.

3.2 Adult Dataset Results

In the case of the Adult dataset we also consider the significance of a smaller block size, thus all properties are expressed in terms of block sizes of 1000 and 5000. From Table 7 it is evident that a smaller block size results in a significant speedup in CPU time to complete a run. Such an observation mirrors that in cache block size design, with larger blocks encountering a higher transfer penalty if this is not leveraged into extended utilization. Relative to the KDD case, an extra five minutes appear to be necessary indicating that although the dataset is smaller, there is more diversity in the dataset, where this is also reflected in the additional complexity in the solutions, Table 7. Detection and FP rates, Table 7, demonstrate variation between RSS-DSS for different block sizes, whereas DSS-DSS retain consistency across the two block sizes.

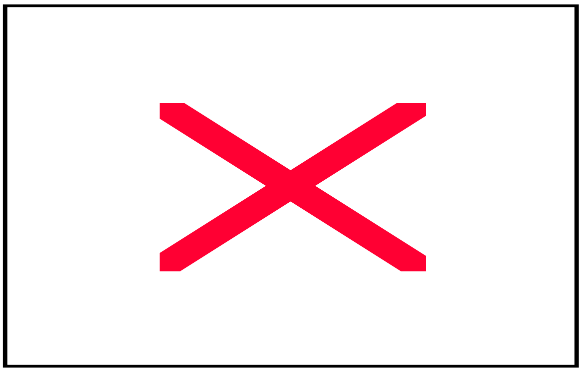


Fig. 3. User to Root (U2R) Accuracy.

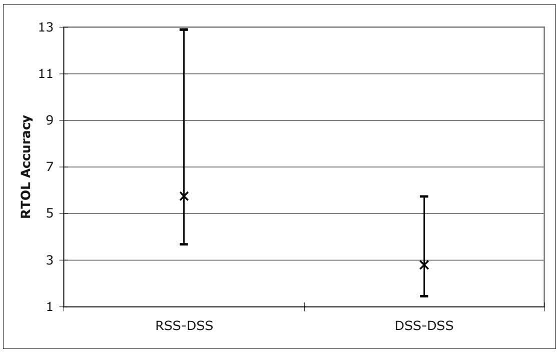


Fig. 4. Remote to Local (R2L) Accuracy.

Table 6. Comparison with KDD-99 winning entries

Parameter	Detection Rate	FP Rate
Winning Entry	0.908819	0.004472
Second Place	0.915252	0.00576
Best GP (RSS-DSS)	0.889609	0.0128108
Best GP (DSS-DSS)	0.897184	0.0062568

Moreover, median Detection rates for DSS-DSS exceed that for RSS-DSS and pair wise comparison of median FP rates also prefers the DSS-DSS algorithm. In terms of solution complexity, there is no significant difference between the two algorithms, but a smaller block size appears to result in less complex solutions, Table 7. Thus, the optimal parameterization appears to be in the smaller block size – faster training time and simpler solutions – with the DSS-DSS algorithm.

In order to provide some qualification of the classifier performance Table 8 details (best case) error rates of alternative machine learning algorithms summarized as part of the UCI repository [15]. Unfortunately no information is provided regarding the

Table 7. Performance of RSS-DSS and DSS-DSS on Adult Dataset

Algorithm	RSS-DSS		DSS-DSS	
Block Size	1,000	5,000	1,000	5,000
	Run Time (minutes)			
Median	10.32	18.46	11.53	15.97
	Detection Rate			
1st Quartile	0.8129	0.8012	0.8254	0.8061
Median	0.8446	0.8253	0.8575	0.8537
3rd Quartile	0.9237	0.8471	0.9122	0.9013
	False Positive Rate			
1st Quartile	0.2600	0.2302	0.2962	0.2597
Median	0.3300	0.2857	0.3549	0.3185
3rd Quartile	0.5403	0.3643	0.4630	0.4489
	Solution Complexity – Before Intron Removal			
Median	127	179	143	171
	Solution Complexity – After Intron Removal			
Median	62	88	60	83.5

learning algorithms, the reliability with which these results are attained or any pre-processing performed to achieve these results. Table 9 lists the error rates of the best-case RSS-DSS / DSS-DSS algorithms with block sizes of 5000, 1000, 500 and 250. It is readily apparent that the GP solutions are ranked towards the end of the list, however, they also appear before the step change in errors from an error of 17 to an error of 19.5. Moreover, no attempt was made to optimize parameters such as the ratio between difficulty and age (boosting algorithms in wide spread use are based on difficulty alone). We also note that as block size decreases, error rates also decrease whilst the preference for the DSS-DSS algorithm becomes more apparent. Moreover, additional experiments with the C5.0 tree induction algorithm indicated that high classification accuracies are indeed reliably achieved (test set error rate of 14.6%), albeit at the expense of solution complexity (over 430 nodes, verses 60-170 instructions in GP). Thus, GP solutions appear to be trading complexity for precision under the parameterization considered here.

One final comparison is made. In this case results are reported for a vanilla implementation of Koza's original Tree structured GP [13] using lilgp [19]. This provides GP baseline performance figures for classification accuracy, run time and solution complexity. Lilgp represents an efficient C code implementation of Tree-structured GP and is parameterized using the default values of: Pop. Size 4000; Crossover prob. 90%; Reproduction prob. 10%; Mutation prob. 0%. As per the page-based linear GP results a node limit of 256 nodes was utilized (as opposed to a depth limit). Results are reported in terms of median and best case error and CPU time for a total of 10 trials in Table 10. It is now apparent that the DSS-DSS algorithm has not negatively impacted on the classification performance whilst retaining a computational speedup of 4 orders of magnitude.

Table 8. Error Rates on Adult Dataset

Algorithm	Error	Algorithm	Error
FSS Naïve Bayes	14.05	Voted ID3 (0.6)	15.64
NBTree	14.10	CN2	16.00
C4.5-auto	14.46	Naïve-Bayes	16.12
IDTM (decision table)	14.46	Voted ID3 (0.8)	16.47
HOODG	14.82	T2	16.84
C4.5 rules	14.94	1R	19.54
OC1	15.04	Nearest-neighbor (3)	20.35
C4.5	15.54	Nearest-neighbor (1)	21.42

Table 9. RSS-DSS and DSS-DSS Error Rates on Adult Dataset

Block Size	5000	1000	500	250
RSS-DSS	17.56	16.78	16.95	16.95
DSS-DSS	17.07	16.95	16.63	16.46

Table 10. Tree Structured GP on Adult Dataset, no RSS/DSS algorithm

CPU time (hrs)	% train	% test	Best error
16.3	82.95	82.91	16.65

4 Conclusion

The computational overhead of the GP inner loop is addressed by introducing a hierarchy of training subset selections. This enables the scaling up of the size of problems for which approaches based on GP may be applied. To do so, the original problem is divided into a series of blocks. Blocks are either selected uniformly (RSS) or relative to their age and difficulty (DSS). Exemplars are sampled from a block relative to their relative block age and difficulty (DSS). Such a scheme matches the observations used to formulate the memory hierarchy typically employed in computer architectures. The ensuing algorithm, DSS-DSS, appears to be competitive with alternative learning algorithms previously applied, with the advantage that the computational overhead of GP is significantly reduced. The method differs from alterative sampling algorithms such as ‘boosting’ by introducing the concept of age and difficulty (boosting is explicitly based on exponentially weighted difficulties) and utilizing a hierarchy of samples. The latter point is fundamental in efficiently scaling the algorithm to larger datasets than is normally possible without specialist hardware. That is to say, competitive solutions are located in minutes rather than hours, and the framework is not specific to GP, thus potentially applicable to other learning algorithms such as neural networks.

Acknowledgements. The authors acknowledge the support of NSERC Discovery, IRIS Emerging Opportunities (graduate scholarship) and CFI New Opportunities (infrastructure) Grants.

References

1. Bennett III F.H. et al.: Building a Parallel Computer System for $18,000 that Performs a Half Petra-Flop per Day. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), Morgan Kaufmann (1999) 1484-1490
2. Juillé H., Pollack J.B.: Massively Parallel Genetic Programming. In: Angeline P.J., Kinnear K.E. (eds): Advances in Genetic Programming 2, Chapter 17. MIT Press, Cambridge, MA (1996) 339-358
3. Koza J.R. et al.: evolving Computer Programs using Reconfigurable Gate Arrays and Genetic Programming. Proceedings of the ACM 6th International Symposium on Field Programmable Gate Arrays. ACM Press. (1998) 209-219
4. Breiman L.: Bagging predictors. Machine Learning. 24(2) (1996) 123-140
5. Freund Y., Schapire R.E.: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of computer and Systems Sciences. 55 Academic Press. (1997) 119-139
6. Hennessy J.L., Patterson D.A.: Computer Architecture: A Quantitative Approach. 3rd Edition. Morgan Kaufmann, San Francisco, CA (2002)
7. Gathercole C., Ross P.: dynamic Training Subset Selection for Supervised Learning in Genetic Programming. Parallel Problem Solving from Nature III. Lecture Notes in Computer Science. Vol. 866. Springer Verlag. (1994) 312-321
8. Song D., Heywood M.I., Zincir-Heywood A.N.: A Linear Genetic Programming Approach to Intrusion Detection. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO). Lecture Notes in Computer Science. Vol. 2724 Springer-Verlag. (2003) 2325-2336
9. Cramer N.L.: A Representation for the Adaptive Generation of Simple Sequential Programs. Proceedings of the International Conference on Genetic Algorithms and Their Application (1985) 183-187
10. Nordin P.: A Compiling Genetic Programming System that Directly Manipulates the Machine Code. In: Kinnear K.E. (ed.): Advances in Genetic Programming, Chapter 14. MIT Press, Cambridge, MA (1994) 311-334
11. Huelsbergen L.: Finding General Solutions to the Parity Problem by Evolving Machine-Language Representations. Proceedings of the 3rd Conference on Genetic Programming. Morgan Kaufmann, San Francisco, CA (1998) 158-166
12. Heywood M.I., Zincir-Heywood A.N.: Dynamic Page-Based Linear Genetic Programming. IEEE Transactions on Systems, Man and Cybernetics – PartB: Cybernetics. 32(3) (2002), 380-388
13. Koza J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA (1992)
14. Elkan C.: Results of the KDD'99 Classifier Learning Contest. SIGKDD Explorations. ACM SIGKDD. 1(2), (2000) 63-64
15. UCI Machine Learning Repository. (2003) http://www.ics.uci.edu/~mlearn/MLRepository.html
16. Lichodzijewski P., Zincir-Heywood A.N., Heywood M.I.: Host-Based Intrusion Detection Using Self-Organizing Maps. IEEE-INNS International Joint Conference on Neural Networks. (2002) 1714-1719
17. Brameier M., Banzhaf W.: A Comparison of Linear genetic Programming and Neural Networks in Medical data Mining. IEEE Transaction on Evolutionary Computation. 5(1) (2001) 17-26
18. Chittur A.: Model Generation for Intrusion Detection System using Genetic Algorithms. http://www1.cs.columbia.edu/ids/publications/ (2001) 17 pages
19. Punch B., Goodman E.: lilgp Genetic Programming System, v 1.1. http://garage.cps.msu.edu/software/lil-gp/lilgp-index.html (1998).

A Multi-objective Genetic Algorithm Based on Quick Sort

Jinhua Zheng[1, 3], Charles Ling[2], Zhongzhi Shi[3], Juan Xue[1], and Xuyong Li[1]

[1] College of Information Engeering, Xiangtan University, Hunan, China
[2] Department of Computer Science, University of Western Ontario, London, Canada
[3] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
jhZheng@xtu.edu.cn, cling@csd.uwo.ca, shiZZ@ics.ict.ac.cn

Abstract. The Multi-objective Genetic Algorithms (MOGA) based on Pareto Optimum have been widely applied to solve multi-objective optimal problems, mainly because of their ability to find a set of candidate solutions within a single run. In MOGAs, a non-dominated set is a set of candidate solutions, so it is very important to construct the non-dominated set efficiently. In this paper, the relation of individuals and their related features are discussed. It is proved that the individuals of an evolutionary population can be sorted by quick sort. We construct the non-dominated set of the evolutionary population with quick sort, and the time complexity of the construction is O($n\log n$), compared to the previous best result of O(n^2) described in the popular NSGA-II [Deb, 2002]. We further propose a multi-objective genetic algorithm based on quick sort, and two benchmark problems are experimented. We show that the results of the experiments match to our theoretical analysis.

1 Introduction

Since Vector Evaluated Genetic Algorithm in machine learning [Schaffer, 1984] and multi-objective optimization using Genetic Algorithms [Goldberg, 1989] were proposed, many researchers have been interested in Multi-Objective Genetic Algorithm (MOGA) or Multi Objective Evolutionary Algorithm (MOEA). This is mainly because MOGAs have the ability to solve complicated tasks such as discontinuous, multi-modal and noisy function evaluations and to find a widespread of Pareto optimal solutions in a single run.
There are several kinds of MOGAs, and here we classify them into two classes: one with parameters and the other with no parameters. Most of the research of MOGAs, especially the earlier studies, focused on the methods with parameters. The main merit of MOGAs with parameters is more efficient than that with no parameters [Fonseca, 1993; Horn 1993, 1994]. There are also some distinct disadvantages such as premature convergence because of the select pressure, and difficulty to select parameters and to modify parameters dynamically [Goldberg, 1991]. Hence, some scientists suggested another kind of MOGAs with no parameters.

The two popular MOGAs with no parameters are NSGA and SPEA. Zitzler and Thiele [1999] proposed Strength Pareto Evolutionary Algorithm (SPEA), in which the fitness of individuals is called strength. The time complexity of calculating the fitness of individuals is O(n^3). Zitzler [2001] improved SPEA, and advanced it to SPEA2 in which the time complexity is O(n^2log$n$). The truncation procedure is adopted to keep the diversity of the candidate solutions, which resembles the crowding procedure in NSGA. Srinivas and Deb [1994] suggested Non-dominated Genetic Algorithm (NSGA), in which a set of candidate solutions, called a non-dominated set, is constructed. The time complexity to construct a non-dominated set is O(n^2). The NSGA keeps its diversity by a crowding procedure in which the distance between two consecutive individuals is calculated. On the basis of NSGA, Deb [2001] proposed a new method — NSGA-II, and a clustering procedure is adopted to keep the diversity of the candidate solutions. Nowadays, the NSGA-II and SPEA2 are very popular in the field of the MOGA research. The MOGAs with no parameters overcome the disadvantages in MOGAs with parameters, but they have a distinct disadvantage of low efficiency.

This paper first briefly describes the multi-objective optimization and Pareto optimal solutions (section 2.1). To make MOGAs with no parameters more efficient, we investigate the relation between evolutionary individuals and their related features (section 2.2). It is proved that the individuals of an evolutionary population can be sorted by quick sort (section 2.2). We propose a new method to construct the non-dominated set using quick sort (section 3), and the time complexity of the construction is O($n$log$n$). We suggest a multi-objective genetic algorithm based on quick sort, and a clustering procedure to maintain the diversity of the candidate solutions (section 4.1). Our new MOGA is tested with two benchmark problems in experiments, and the results are shown that the results match to the theoretical analysis (section 4.2).

2 Multi-objective Optimization and Some Features

We discuss the concept of multi-objective optimization and Pareto optimal solution in the first subsection, illustrating with an example of two objectives. Then we present the relation of individuals and some of their features in the second subsection, and we prove that the individuals of an evolutionary population can be sorted by quick sort.

2.1 Multi-objective Optimization

In general, a multi-objective optimization is defined as follows:

Find the vector $X^* = [X_1^*, X_2^*, \cdots, X_n^*]^T$, to satisfy k inequality constraints:

$$g_i(X) \ge 0 \quad i = 1,2,\cdots,k \tag{1}$$

l equality constraints:

$$h_i(X) = 0 \quad i = 1,2,\cdots,l \tag{2}$$

and optimize the vector function:

$$\min \quad f(X) = [f_1(X), f_2(X), \cdots, f_m(X)]^T \tag{3}$$

where $X=[X_1,X_2,\cdots,X_n]^T$ is the vector of decision variables.

We say that a point $X^* \in F$ is Pareto optimal if $\forall X \in F$ either,

$$\bigcap_{i\in I}(f_i(X)=f_i(X^*)) \tag{4}$$

or, there is at least one i$\in I$ such that

$$f_j(X) > f_j(X^*) \tag{5}$$

here $F=\{X\in R^n \mid g_i(X)\geq 0, i=1,2,\cdots,k; h_j(X)=0, j=1,2,\cdots,l\}$, I={1, 2,…,m}.

The concept of Pareto optimum was formulated by Vilfredo Pareto in 1896. In general, there is more than one Pareto optimal solution. The goal of MOGAs is to construct a non-dominated set, and to make the non-dominated set approach the optimal solutions continually until it arrives the optimal solutions at last.

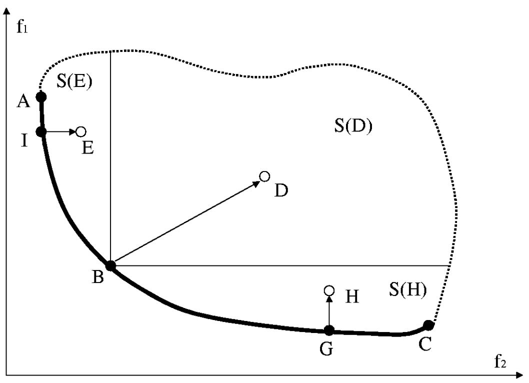


Fig. 1. Pareto Front for two objectives

For simplicity, we use the individuals in vector space (or gene space) to denote their corresponding individuals in objective space (or phenotype space) in Figure 1 and Figure 2. As shown in Figure 1, Pareto Optimal solutions of two conflicting objective functions form a curved bold line. Assume that B is a feasible solution on the bold line. Because there exists no another feasible individuals X (vectors) which would decrease some criterion without causing a simultaneous increase in at least one criterion, B is a Pareto optimal solution. More specifically, we divide F into three parts (see Figure 1): S(D) is a region with individual D in it and includes the boundary, S(E) with individual E in it and not include the boundary, and S(H) with individual H in it and not including the boundary. $\forall$X$\in$ S(D), f_1(B)$\leq$ f_1(X), f_2(B)$\leq f_2$(X), so there exists no feasible X$\in$ S(D) better than B. $\forall$X$\in$ S(E), f_1(B)<f_1(X),f_2(B)>f_2(X), so individual B in one criterion (objective 1) is better than X. As for $\forall$X$\in$ S(H), we have f_2(B)<f_2(X), which means B in objective 2 is better than X. Therefore we can conclude that individual B is one of Pareto optimal solutions, and we can also indicate that every individual (or vector) in bold line is Pareto optimal solution. Then let's discuss the individuals that do not lie on the bold line. Assume individual D∈ S(D), it is easy to find that individual B is better than D, because f_1(B)<f_1(D) and f_2(B)<f_2(D), so individual D is not a Pareto optimal solution. The same is true to individuals E∈ S(E) and H∈ S(H), because f_1(I)=f_1(E) and f_2(I)<f_2(E), f_1(G)<f_1(H) and f_2(G)=f_2(H), so they are not Pareto optimal solutions. Hence, we can conclude that every individual not in the bold line is no Pareto optimal solution.

The above examples explain the cases in equation (5), and the description in equation (4) is a very special case that means all individuals (vectors) with the same function value for each objective exclusively.

2.2 Relation between Individuals and Some Related Features

Before discussing how to use quick sort to construct the non-dominated set, let's define the relation, and then discuss some features on the basis of the definitions. Assuming Pop as an evolutionary population, and vectors X, Y∈ Pop as two individuals of population, there is a relation on whether one dominates the other, or none dominates the other. In other words, there are three cases: the first is that X dominates Y, the second is that Y dominates X, and the third is that X does not dominate Y and Y does not dominate X. we give a definition about "dominate" as follow.

Definition 1: Suppose that Pop is an evolutionary population, ∀X, Y∈ Pop, X is said to dominate Y, if both of the two conditions are true:

(1) Individual X is no worse than individual Y in all objectives, or $f_k(Y) \geq f_k(X)$ for all $k = 1,2,\cdots,m$ objectives;

(2) The individual X is strictly better than individual Y in at least one objective, or $\exists l \in \{1,2,\cdots,m\}$ for $f_l(Y) > f_l(X)$.

We say X is a non-dominated individual, and Y is a dominated individual, or X $\succ$ Y, here "$\succ$" denotes the dominated-relation.

Definition 2: ∀X, Y∈ Pop, we say X relates with Y, iff X$\succ$ Y or Y$\succ$X or X= Y; otherwise we say there is no relation between X and Y.

Lemma 1: individuals in a non-dominated set do not relate with each other.
Proof by contradiction: Let {Y_1, Y_2,..., Y_k} be a non-dominated set, and ∀Y_i, Y_j∈ {Y_1, Y_2, ..., Y_k} be two different non-dominated individuals, here i≠j.

Assume that Y_i relates with Y_j, according to definition 2, and then there are two cases:
(1) Y_i = Y_j, which means the two individuals are the same. This case does not exist.
(2) $Y_i$$\succ$ Y_j, which means Y_i dominates Y_j; or $Y_j$$\succ$ Y_i, which means Y_j dominates Y_i.

From discussion above, we can indicate that there must exist one individual that dominates another. This conclusion is conflict with the assumption, so individual Y_i does not relate with Y_j. Hence we conclude that individuals in a non-dominated set do not relate with each other. (End of proof).

From Lemma 1, we conclude that the individuals in an evolutionary population cannot be sorted in terms of relation "$\succ$", so a new relation must be defined between individuals of an evolutionary population.

Definition 3: $\forall$ X, Y$\in$ Pop $X >_d Y$ iff $X \succ Y$ or X does not relate with Y.

Lemma 2: The relation "$>_d$" is not transitive.

Proof: $\forall$ X, Y, Z$\in$ Pop, suppose $X >_d Y$, $Y >_d Z$. From the definition of "$>_d$", there are four cases:

(1) $X \succ Y$, Y and Z have no relations.

Assume there are r objective functions, then we have:

$X._H \succ Y._H$, $Y._A \succ Z._A$, $Z._B \succ Y._B$ and $A \neq \phi$, $B \neq \phi$, $A \cap B = \phi$, $A \cup B = H$, $|A|+|B|=|H|=r$.

So, $X._A \succ Z._A$ but the relation between $X._B$ and $Z._B$ is indefinite.

Let $B = C \cup D$ and $C \cap D = \phi$ such that $X._C \succ Z._C$ and $Z._D \succ X._D$.

Then $X._{A \cup C} \succ Z._{A \cup C}$ and $Z._D \succ X._D$, i.e., $X >_d Z$. When $D = \phi$, we have $X \succ Z$.

(2)There are no relations between X and Y, but $Y \succ Z$.

Suppose that there are r objective functions.

Then $X._A \succ Y._A$, $Y._B \succ X._B$, $Y._H \succ Z._H$, $A \neq \phi$ and $B \neq \phi$, $A \cap B = \phi$, $A \cup B = H$, $|A|+|B|=|H|=r$.

that is $X._A \succ Z._A$, however, the relation between $X._B$ and $Z._B$ is not definite.

Let $B = C \cup D$ and $C \cap D = \phi$ such that $X._C \succ Z._C$ and $Z._D \succ X._D$.

It follows easily that, $X._{A \cup C} \succ Z._{A \cup C}$ and $Z._D \succ X._D$. So, $X >_d Z$.

When $D = \phi$, we have $X \succ Z$.

(3)There are no relations between X and Y, nor Y and Z.

Suppose that there are r objective functions.

Then $X._A \succ Y._A$, $Y._B \succ X._B$, $A \neq \phi$, $B \neq \phi$, $A \cap B = \phi$, $A \cup B = H$, and $|A|+|B|=|H|=r$.

Also $Y._C \succ Z._C$, $Z._D \succ Y._D$, $C \neq \phi$, $D \neq \phi$, $C \cap D = \phi$, $C \cup D = H$, and $|C|+|D|=|H|=r$.

In other words, $X._{A \cap C} \succ Z._{A \cap C}$, $Z._{B \cap D} \succ X._{B \cap D}$,

but the relations between $X._{H-(A \cap C)-(B \cap D)}$ and $Z._{H-(A \cap C)-(B \cap D)}$ are not definite.

Let $H-(A \cap C)-(B \cap D) = E \cup F$ and $E \cap F = \phi$ such that $X._E \succ Z._E$ and $Z._F \succ X._F$.

It follows that $X._{(A \cap C) \cup E} \succ Z._{(A \cap C) \cup E}$ and $Z._{(B \cap D) \cup F} \succ X._{(B \cap D) \cup F}$.

If $(A \cap C) \cup E$ is not empty, then $X >_d Z$; If $(A \cap C) \cup E = \phi$, then $Z \succ X$.

Therefore, "$>_d$" is not transitive in this case.

(4)$X \succ Y$, $Y \succ Z$.

By the definition of "$\succ$", it is clear that X$\succ$ Z in this case.

Therefore, from the discussion above, it is conclude that the relation "$>_d$" is not transitive.

(End of proof)

It is clear from the Lemma 2 that sort procedures, such as merge sort, heap sort, and tree type sort, which request the sorting elements to have transitivity, can't be used to sort individuals in terms of relation "$>_d$". We can use these sort procedures, such as bubble sort and quick sort, which do not request the sorting elements to have transitivity, to construct a non-dominated set from population in terms of relation "$>_d$".

3 Construct Non-dominated Set Using Quick Sort

In SPEA2, the fitness of individuals can be calculated as follows: $F(i)=R(i)+D(i)$; here $R(i)$ denotes the sum of individuals that are dominated by these individuals dominating i. $D(i)=1/(\sigma_i^k+2)$, $k=\sqrt{n}$, n is the size of evolutionary population, and σ_i^k denotes the distance from i to k^{th} neighborhood of i. The time complexity of calculating the fitness of individuals is O(n^2log$n$) [Zitzler, 2001]. NSGA-II calculates n_i and s_i of each individual by a nested loop, where n_i records the sum of individuals that dominate individual i, and s_i is a set of individuals that are dominated by i. Then the algorithm classifies the population by formula P_k={ i | n_i-k+1=0}, where P_k is a non-dominated set when k=1, and a dominated set when k>1. The time complexity to construct a non-dominated set is O(n^2) [Deb, 2002].

In this section, we study how to construct a non-dominated set with quick sort, and the time complexity of the construction is O($n$log$n$). For each step, an individual X is selected from the population as a reference individual (in general, the first one selected), and every other individual of the population is compared with X in turn. After a round of comparison, all the individuals of the population are divided into two subsets. The first subset contains individuals dominated by X and will be discarded in the next step, while the second subset contains individuals dominating X or unrelated with X. If X is not dominated by any individual of the second subset, X is a non-dominated individual of the population and merged into the non-dominated set. If X is dominated by any individual of the second subset, X will be also discarded in the next step. The process will be repeated until there is only one individual in the second subset.

In the quick sort procedure, when none of the individuals from the second subset {X_1, X_2, …, X_k} relates with X, it is certain that X is a non-dominated individual of the population, but it is not certain that individual from the second subset {X_1, X_2, …, X_k} is a non-dominated individual of the population, especially in early stage of the sort procedure. For the example shown in Figure 2, assume β={A, B, C, D, E, F, G,

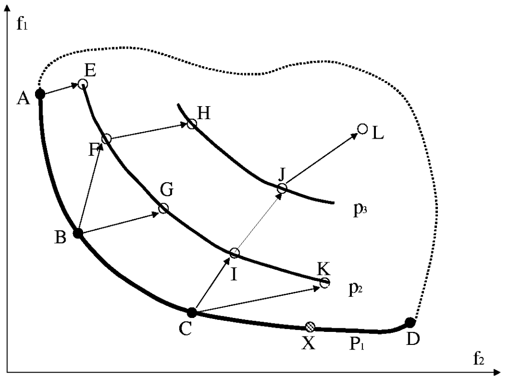


Fig. 2. Relationships among individuals

H, I, J} $\subseteq$ {X_1, X_2, …, X_k}, and X does not relate with any member of β. Because there exist the following relations: A $\succ$ E, B $\succ$ { E, F, G, H}, C $\succ$ {H, I, J}, G $\succ$ {H, J}, and I $\succ$ J, it is obvious that some individuals from β are dominated individuals, such as {E, F, G, H, I, J}.

Theorem: assume Pop={Y_1, Y_2, …, Y_m}, $\forall$ X$\in$ Pop, Pop is divided into two parts by X: {X_1, X_2, …, X_k} and {X_{k+2}, X_{k+3}, …, X_m}. if $\forall$ Y$\in$ {X_1, X_2, …, X_k}, $\forall$ Z$\in$ {X_{k+2}, X_{k+3}, …, X_m}, $Y >_d X$ and X $\succ$ Z, then:
(1) $\forall$ Y$\in$ {X_{k+2}, X_{k+3}, …, X_m} is a dominated individual;
(2) if $\forall$ Y$\in$ {X_1, X_2, …, X_k}, X is not dominated by Y, or $\neg$(Y $\succ$ X), then X is a non-dominated individual of Pop;
(3) $\forall$ Y$\in$ {X_1, X_2, …, X_k}, if Y is a non-dominated individual of {X_1, X_2, …, X_k}, then Y is also a non-dominated individual of Pop.
Proof:
(1) It is obvious that Y is dominated by X, because $\forall$ Y$\in$ {X_{k+2}, X_{k+3}, …, X_m} , we have X $\succ$ Y. It means that Y is a dominated individual.
(2) Because $\forall$ Y$\in$ {X_1, X_2, …, X_k}, X is not dominated by Y, or $\neg$(Y $\succ$ X), and $\forall$ Z$\in$ {X_{k+2}, X_{k+3}, …, X_m}, X $\succ$ Z; it is concluded that X is not dominated by v, $\forall$ v$\in$ {X_1, X_2, …, X_k} $\cup$ {X_{k+2}, X_{k+3}, …, X_m}. So, it means that X is a non-dominated individual of Pop;
(3) First, $\forall$ Y$\in$ {X_1, X_2, …, X_k}, Y is not dominated by any other individuals from {X_1, X_2, …, X_k}, because Y is a non-dominated individual of {X_1, X_2, …, X_k}. Second, because $Y >_d X$, it means that Y is not dominated by X, and $\forall$ Z$\in$ {X_{k+2}, X_{k+3}, …, X_m}, X $\succ$ Z, it means Y is not dominated by Z. So, it is concluded that Y is not dominated by anyone from {X_1, X_2, …, X_k} $\cup$ {X} $\cup$ {X_{k+2}, X_{k+3}, …, X_m}, i.e., the individual Y is a non-dominated individual of Pop. (End of Proof).

Using this Theorem, we can construct a non-dominated set with quick sort, as shown in algorithm 1.

Algorithm 1: construct a non-dominated set with quick sort

```
Function NDSet (Var Pop: Evolutionary Population; s, t: integer)
{ NDSet = $\phi$;
  i=s; j=t; X=Pop[s];
  while |Pop|>2 do
  { non-dominated-sign=.T.;
     while (i < j ) do
     { while (i < j ) and (X $\succ$ Pop[j]) do j=j-1;
       p=Pop[i]; Pop[i]=Pop[j]; Pop[j]=p;
         while (i < j ) and (Pop[j]$>_d$ X) do
           { i=i+1; if (Pop[j] $\succ$ X) then non-dominated-sign=.F. }
       p=Pop[i]; Pop[i]=Pop[j]; Pop[j]=p;
     }
     if (non-dominated-sign) then NDSet = NDSet $\cup$ {X};
     j=i-1; i=s; X=Pop[s];
  }
}
```

The function NDSet( ) returns a non-dominated set of population. It can be proved that the time complexity of algorithm 1 is less than O($n$log$n$). It is better than O(n^2) in Deb [2002].

4 Multi-objective Genetic Algorithm Based on Quick Sort

In this section, we describe a multi-objective genetic algorithm that based on Quick Sort in the first subsection. Then we test our MOGA with two benchmark problems in experiments in the second subsection.

4.1 Multi-objective Genetic Algorithm

In this subsection, we will discuss some details about the MOGA based on quick sort. Suppose that the size of the evolutionary population Pop is n, and Pop_t is tth generation of the population. Q_t is a new evolutionary population from Pop_t that is updated by the selection, crossover and mutation operators, and the size of Q_t is also n. Let $R_t=Pop_t \cup Q_t$, and the size of R_t is 2n. The non-dominated set P_1 is generated from R_t with the quick sort procedure.

If $|P_1|<n$, then $(n-|P_1|)$ individuals are generated with a clustering procedure [Zitzler, 1999] from (R_t-P_1), which constitutes the new evolutionary population P_{t+1} with P_1.

If $|P_1|>n$, the clustering procedure, crowding procedure [Deb 2002, 2003], or truncation procedure [Zitzler, 2001] is used to reduce the size of P_1, and to keep the diversity of P_1 at the same time. The size of P_1 will be n after the reducing process.

All the procedures are briefly shown in algorithm 2.

Algorithm 2: a MOGA based on quick sort

```
Quick-Sort-MOGA($Pop_0$)
{ $Q_t = \psi(Pop_t)$; //here $\psi$ including selection, crossover and mutation operators
    $R_t = Pop_t \cup Q_t$; // merge the last population with the current new population
   $P_1$=NDSet($R_t$); //construct the non-dominated set $P_1$ with quick sort
   if ($|P_1|<n$)
        then $Pop_{t+1}= P_1 \cup$ Cluster ($R_t-P_1$, $n-|P_1|$)
     else if ($|P_1|>n$) then $Pop_{t+1}$ =Cluster ($P_1$, n);
        //here Cluster ($P_1$, n) can be replaced with Crowd ($P_1$, n) or with Truncate($P_1$, n)
   t=t+1;
}
```

Here Cluster(w, m) is a clustering procedure, w is a population on which the clustering operator applies, and m is the size of the result of clustering. By executing the procedure Cluster (w, m), m individuals are selected from the set w, and the diversity of the set w can be kept after selecting operator. The other two procedures Crowd (w, m) and Truncate (w, m) have the same function as Cluster (w, m) in terms of reducing the size of a non-dominated set.

4.2 Experiments

In this subsection, we apply the MOGA based on quick sort (QKMOGA) to two benchmark test problems. The first one is a two objective problem MOP4, and the second one is a three objective problem DLTZ2. The description of MOP4 is as follows:

$$MOP4: \begin{cases} f_1(x) = \sum_{i=1}^{n-1}(-10\exp(-0.2\sqrt{x_i^2 + x_{i+1}^2})) \\ f_2(x) = \sum_{i=1}^{n}(|x_i|^{0.8} + 5\sin(x_i)^3) \end{cases} \qquad -5 \le x_1, x_2, x_3 \le 5$$

We compare QKMOGA with the other two popular MOGAs: NSGA-II and SPEA2. In the two experiments, the three kinds of MOGAs have the same computational environment of a Pentium 4 PC with 1.7GHZ, 256M DDR RAM, Windows 2000, and visual C++ 5.0. The binary code is adopted in all the three algorithms, the size of population is 100, and the probability of mutation is 1/len (len is the length of code). The results shown in Table 1 are the average of 10 runs, and each run is of 100 generations. There are two cases: one is to use the crowding procedure to reduce the size of the non-dominated set, and the other is to use the clustering procedure. It is shown in Table 1 that QKMOGA is faster than the other two algorithms whether to use the crowding procedure or the clustering procedure. When using the crowding procedure, each algorithm is faster than that by using the clustering procedure. Note that the time unit is second (s) in Table 1 and Table 2.

Table 1. CPU-time comparison in a two objective problem

	QKMOGA	NSGAII	SPEA2
Crowding procedure	0.8552s	1.1663s	1.2538s
Clustering procedure	0.9367s	1.3254s	1.4673s

The description of DLTZ2 is as follows:

$$\begin{aligned}
&\textit{Minimize} \quad f_1(X) = (1+g(X_K))\cos(x_1\pi/2)\cos(x_2\pi/2)\cdots\cos(x_{m-2}\pi/2)\cos(x_{m-1}\pi/2),\\
&\textit{Minimize} \quad f_2(X) = (1+g(X_K))\cos(x_1\pi/2)\cos(x_2\pi/2)\cdots\cos(x_{m-2}\pi/2)\sin(x_{m-1}\pi/2),\\
&\textit{Minimize} \quad f_3(X) = (1+g(X_K))\cos(x_1\pi/2)\cos(x_2\pi/2)\cdots\sin(x_{m-2}\pi/2),\\
&\qquad\qquad\quad \cdots\cdots\\
&\textit{Minimize} \quad f_{m-1}(X) = (1+g(X_K))\cos(x_1\pi/2)\sin(x_2\pi/2),\\
&\textit{Minimize} \quad f_m(X) = (1+g(X_K))\sin(x_1\pi/2),
\end{aligned}$$

$$0 \le x_i \le 1, \; \textit{for} \quad i = 1,2,\ldots,n$$

where $g(X_K) = \sum_{i=m}^{n}(x_i - 0.5)^2, \; (x_m,\cdots,x_n) \in X_K$

$$X = \{(x_1, x_2, \cdots, x_{m-1}, x_m, \cdots x_n)\}, X_K = \{(x_m, \cdots, x_n)\}$$

$u=|X_K|=10$, $m=3$, $n=u+m-1=12$.

The results shown in Table 2 are the results of 300 generations. It is shown in Table 2 that QKMOGA is faster than the other two algorithms whether to use the truncation procedure or the clustering procedure. When using the truncation procedure, each algorithm is faster than the ones that use the clustering procedure.

Table 2. CPU-time comparison in a three objective problem

	QKMOGA	NSGAII	SPEA2
Truncation procedure	3.624s	5.346s	5.823s
Clustering procedure	3.892s	5.921s	6.217s

In the two experiments, all three algorithms are convergent to the true Pareto optimal front and the non-dominated sets show a good diversity. As our main topic in this paper is to discuss the time efficiency, we will not discuss details on the convergence and diversity of the MOGAs, which can be found in [Deb and Jain, 2002; Khare, 2002].

5 Conclusion

In this paper, we describe the multi-objective optimization problem and the Pareto optimal solutions. We give the formal definition of the dominated relation between two individuals. It is proved that individuals in a non-dominated set do not relate with each other, so the individuals cannot be sorted in terms of that relation. Therefore, a new relation is defined. It is proved that the individuals of evolutionary population can be sorted with quick sort in terms of the new relation. The time complexity to construct a non-dominated set is O($n$log$n$), which is better than O(n^2) in NSGA-II (Deb, 2002). We suggest a multi-objective genetic algorithm based on quick sort (QKMOGA), and the experiment of two-benchmark testing problems show that the results match to the theoretical analysis.

Acknowledgments. This work presented here is supported in part by the Natural Science Foundation of Chinese Government and in partly by the Natural Science Foundation of the Hunan Province. The authors would like to thank the anonymous reviewers for their constructive comments to help us improve the paper.

References

[Deb, 1999] Kalyanmoy Deb. Evolutionary Algorithms for Multi-Criterion Optimization in Engineering Design, In Kaisa Miettinen, Marko M. Mäkelä, Pekka Neittaanmäki, and Jacques Periaux, editors, *Evolutionary Algorithms in Engineering and Computer Science*, chapter 8, pages 135-161. John Wiley & Sons, Ltd, Chichester, UK, 1999.

[Deb, 2000]Deb K., Agrawal S., Pratap A., & Meyarivan T. (2000). A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL Report No.200001.

[Deb and Jain, 2002]Deb, K and Jain, S. (May, 2002). Running performance metrics for evolutionary multi-objective optimization. *Kangal Report No. 2002004.*

[Deb, 2002]Kalyanmoy Deb, Amrit Pratap, Sameer Agrawal and T. Meyrivan. A Fast and Elitist Multi-objective Genetic Algorithm : NSGA-II. IEEE Transactions on Evolutionary Computation, 6(2):182-197, April 2002.

[Deb, 2003]Deb, K, Mohan, M. and Mishra, S. (February, 2003). A Fast Multi-objective Evolutionary Algorithm for Finding Well-Spread Pareto-Optimal Solutions. *KanGAL Report No. 2003002*.

[Fonseca, 1993] Carlos M. Fonseca and Peter J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization, In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416-423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

[Goldberg, 1989]D. E. Goldberg. Genetic algorithms for search, optimization, and machine learning. Reading, MA: Addison-Wesley, 1989.

[Goldberg, 1991]Goldberg, D. E. and Deb, K. (1991). A comparison of selection schemes used in genetic algorithms, Foundations of Genetic Algorithms, 69-93.

[Horn, 1993] Jeffrey Horn and Nicholas Nafpliotis. Multiobjective Optimization using the Niched Pareto Genetic Algorithm, Technical Report IlliGAl Report 93005, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1993.

[Horn, 1994]Horn J., Nafpliotis N., & Goldberg D. E. (1994). A Niched Pareto genetic Algorithm for Multiobjective Optimization. Proceeding of the first IEEE Conference on Evolutionary Computation, 82-87.

[Schaffer, 1984] J. D. Schaffer. Some experiments in machine learning using vector evaluated genetic algorithms. PhD thesis, Vanderbilt University, 1984.

[Khare, 2002]V. Khare, X. Yao, and Deb, K.. (October, 2002). Performance Scaling of Multi-objective Evolutionary Algorithms. *Kangal Report No. 2002009.*

[Srinivas, 1994] N. Srinivas and Kalyanmoy Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms, *Evolutionary Computation*, 2(3):221-248, Fall 1994.

[Zitzler, 1999]Zitzler, E. and L. Thiele (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation.

[Zitzler, 2001]E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. EUROGEN 2001 - Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems, September 2001, to appear.

Knowledge-Rich Contexts Discovery

Caroline Barrière

Research Center for Language Technology, National Research Center of Canada,
Gatineau, Québec, Canada[1]
Caroline.Barriere@nrc-cnrc.gc.ca

Abstract. Within large corpora of texts, Knowledge-Rich Contexts (KRCs) are a subset of sentences containing information that would be valuable to a human for the construction of a knowledge base. The entry point to the discovery of KRCs is the automatic identification of Knowledge Patterns (KPs) which are indicative of semantic relations. Machine readable dictionary serves as our starting point for investigating the types of knowledge embodied in definitions and some associated KPs. We then move toward corpora analysis and discuss issues of generality/specificity as well as KPs efficiency. We suggest an expansion of the lexical-syntactic definitions of KPs to include a semantic dimension, and we briefly present a tool for knowledge acquisition, SeRT, which allows user such flexible definition of KPs for automatic discovery of KRCs.

1 Introduction

Texts, corpus of texts, are invaluable sources of information. But even if text has the quality of being rich and abundant, it has the default of being unstructured, sequential access, uneven in its "valuable information" rate, disorganized, ambiguous, sometimes redundant or even contradictory. This suits humans just fine when reading casually, but is a large obstacle to any attempt to automatic construction of domain model from text. If we settle for a slightly less ambitious goal of semi-automatic acquisition of information from texts, we still face the challenge of identifying the subset of sentences that contain valuable information.

The notion of value of information is quite subjective and is dependent on the task envisaged. For example, the construction of a concept dictionary [1] to help in question answering does not emphasize the same information as the construction of a Terminological Knowledge Base to help terminographers generate term definitions. As our research over the past years has been closer to the Computational Terminology community, we will adopt that view point in our investigation. Being aware of the large interest in knowledge discovery by the Information Extraction (IE) community, we will also include some comments relating to IE throughout the article.

Meyer [2] refers to those valuable sentences as Knowledge Rich Contexts (KRCs). Not only would a KRC contain a term of interest in a particular domain, but

[1] This research has been performed while the author was at the School of Information Technology and Engineering, University of Ottawa.

it would also contain a Knowledge Pattern (KP) showing how to link it to other terms of the domain. Knowledge Patterns (KPs) are the key to the discovery of Knowledge-Rich Contexts, so that the terminographer need not to look at thousands of sentences in an attempt to define a term, but can look at only the subset of sentences that contain the needed information.

We suggest, in this research, to deepen our understanding of Knowledge Patterns (KPs). Although they are commonly used toward knowledge discovery, although lists are made, although variations are searched for, they are not often discussed with regards to their definition and value. To achieve our goal, we decided to go back to the source and look into dictionary definitions for the types of knowledge embodied and their expression via KPs. This will be presented in Section 2.

In Section 3, we move from dictionary to real text corpora. Although research on dictionary can inspire and guide research on real text, it is not obvious it entirely reflects the type of knowledge necessary to understand a particular domain. Focusing on a specific domain of scuba-diving, looking at a 1M word corpus, we will address questions of generality/specificity of semantic relations and associated KPs.

In Section 4, in light of many examples of KPs presented in Section 2 & 3, we will suggest to expand the usual lexical-syntactic view of KPs to include a semantic dimension. As a KP is an entry point into a text, we want to be able to define it in a flexible way. This section will also explore the notion of productivity for KPs.

Section 5 briefly presents SeRT (Semantic Relations in Text) which allows for the definition of KPs and their use toward the discovery of Knowledge Rich Contexts.

Section 6 concludes.

2 Knowledge Patterns and Types of Knowledge

This section gives a possible organization of the type of information contained in dictionary definitions. This will serve for our discussion on corpora as we come back on the application-dependent notion of valuable information in Section 3.

The predominant relation found in a dictionary is the hyperonymy. Defining a word via its genus (superclass) dates back to Aristotle and persists today. The hyperonymy is part of the set of paradigmatic relations, which includes also synonyms and antonyms. We refer to this paradigmatic view of the conceptual model as **static knowledge**.

Table 1. Paradigmatic relations and their KPs

Semantic Relation	Example from AHFD
Opposite	Back **is the opposite of** front.
Synonymy	Automobile **is another word for** car.
	Earth **means** dirt.
Hyperonymy	An acorn **is a** nut that grows into an oak tree.
	An apple **is a kind of** fruit.
	Dogs, cats**,** birds**, and** insects **are all** animals.

This type of knowledge is very much present in dictionary definitions. Table 1 presents some examples of these paradigmatic relations with some typical KRCs taken from the American Heritage First Dictionary (AHFD).[2]

In fact, the static knowledge encompasses more than paradigmatic relations. We see it as knowledge which does not rely on any external events, which means that it is pretty impermeable to context[3]. Table 2 shows that this type of knowledge is also present in the AHFD, and it also suggests an organization of this knowledge.

Table 2. Static knowledge found in AHFD, categorized + KPs

Know. Type	Semantic Relation	Example
Composition	Part-of	An arm **is a part of** the body.
	Piece-of	A block **is a piece of** wood.
	Area-of	A beach **is an area of** sand.
	Amount-of	A breath **is an amount of** air.
Member-Set	Group	An army **is a large group of** people.
	Member	A letter **is one of** the symbols of the alphabet.
Human/ Animals	Relationship	Your cousin **is the child of** your aunt or uncle.
	Child	A lamb **is a young** sheep.
	Home	A hive **is a** home for bees.
Comparison	Like	A brush **looks like** a small broom.
Description	Name	An address **is the name of** a place.
	Material	Glass is what windows **are made from**.
	Function	A pen **is a tool to** write.
Intrinsic attributes[4]	Color	Toasts and chocolate **are brown**.
	Smell	It (onion) has a **strong smell** and taste.
	Size	A camera is a **small** machine that makes pictures.
	Taste	Lemons have a **sour taste**.

This investigation into the AHFD relates to much previous work on Machine Readable Dictionaries (MRDs). This research question of "how to find interesting knowledge" has been intensively studied, during the years in which doing knowledge acquisition from MRDs was fashionable [3,4,5,6,7]. With corpora now flowing on our desk, such research is often relinquished to the attic. We thought it was a good time to dust it off… Instead of starting from a clean slate when investigating corpora, we can certainly be inspired (even if the context is different) by research done on MRDs.

[2] Copyright @1994 by Houghton Mifflin Company. Reproduced by permission from THE AMERICAN HERITAGE FIRST DICTIONARY.

[3] Permeability to context, even for paradigmatic relations, in our opinion, cannot be absolute. The AHFD defines "prize" as *Prizes can be cups, ribbons, money, or many other things.* The notion of hyponyms (subclasses) of prizes is not really definable. There might be "typical" prizes, but nothing prevents just about anything to be a prize, depending on context.

[4] In Sowa (1984) a distinction is made between (a) attributes, holding for a concept without involving other concepts and (b) relations, which links a concept to other concepts of a domain. Given that distinction, the list given here of "intrinsic attributes" would not be considered a semantic relation.

Many researchers have been looking for KPs in definitions, but have referred to them more as "defining formulae"[5].

In a domain model, objects must be put in dynamic relationship to each other. This includes temporal and causal information. If information about time can be seen as a linear progression of state changes, information about cause-effect introduces a tree-like progression, leading to an immense number of possible state changes, as each node of the tree represents a cause-effect rule that could be apply. The possibility rather than the certainty of applying a rule reveals the intimate link between causality and uncertainty. This interaction between temporality, causality and uncertainty is what we refer to as **dynamic knowledge**.

Although the word dynamic brings to mind actions and therefore verbs, there is still a relation to nouns by the fact that some of them are causal agents. For example, the AHFD's definitions of "dust" and "tornado" involve causality. As such, "Dust can make you sneeze" and "Tornadoes can knock down houses and pull trees out of the ground". These examples relate to causal agent and are part of objects (nouns), somehow given their potential. Looking at verbs, we can call "intrinsic causality" the fact that some verbs intrinsically express transformations, or results or goals. We therefore look into the AHFD to find KPs for intrinsic causality and present our findings in Table 3.

Table 3. Intrinsic causality found in AHFD, categorized + KPs

Semantic Relation	Example
Result	Ash is **what is** left **(pp)** after something burns.
	Smoke **is made by** things that burn.
Cause	To kill **is to cause to** die.
	To pour **is to make** liquid go from one place to another.
	The window broke **when** a baseball went through it.
Transformation	To die means **to become** dead.

Table 4. Intrinsic temporality found in AHFD, categorized + KPs

Semantic Rel.	Example
Non-ordered parts	Exercise is running and jumping and moving your body around. **(list of –ing verbs)**
	The ground shakes and sometimes buildings fall **during** an earthquake.
Time-spanned event	A chase **is when** someone follows something quickly.
	A trip is **a time when** you travel somewhere.
Process	To roll is **to keep** turning over and over.
	To twist is to turn **around and around**.
Sequence	To dip means to put something in liquid **and then** to take it out quickly.

[5] Meyer [1] mentions that KPs have been called formulae, diagnostic frames or test frames, frames, definitional metalanguage and defining expositives, and knowledge probes. We refer the reader to Meyer [1] for all appropriate references.

The second type of dynamic knowledge mentioned above is temporal knowledge. Similarly to the idea of intrinsic causality, we can talk of intrinsic temporality. Some verbs embody within their meaning a succession of events, or minimally the notion that something happening spans over a period of time. We look at KPs to allow us for the discovery of such verbs in Table 4.

We have presented static and dynamic knowledge. A third type of knowledge will be present in a corpus, that of event-related knowledge. The usual who, where, what, when, how, why questions are part of events. An investigation into the AHFD will show that yes, many verbs do carry within their meanings some answers to these questions. Continuing with the intrinsic/extrinsic difference made above, we would say that these verbs contain intrinsic event knowledge. This is different from the long tradition of case role investigation [8], looking at extrinsic event knowledge. Some of these verb definitions are in Table 5.

Table 5. Intrinsic event-related knowledge

Semantic Relation	Example
Instrument	To bite means to cut **with** your teeth.
Method	To blow means to make a sound **by** pushing air.
Manner	To giggle is to laugh **in a** silly **way**.
Direction	To bow means to bend the body **forward**.
Path	To eat means to take food into the body **through** the mouth.
During	To dream means to imagine stories **while** you sleep.
Frequency	To practice is to do something **many times** so that…
Reason	Many people hug each other **to show that** they are glad.
Goal	To chase means to run after something **to try to** catch it.

Furthermore, some nouns are defined as place, agent or point in time. Table 6 shows a few examples. The noun definition therefore leads to a typical action and becomes an answer to who, where or when for that action.

Table 6. Intrinsic event-related knowledge expressed by nouns

Semantic Relation	Example
Agent	A barber **is a person who** gives haircut.
Location	An airport **is a place where** airplanes take off and land.
Point-in-Time	Birth is **the moment when** a person is born.

In conclusion, we have found in the definitions of a dictionary (which has traditionally been the knowledge repository for humans) three types of knowledge: (a) static knowledge of which the important fragment of paradigmatic relations is well known and well used in knowledge bases (b) dynamic knowledge comprising causal and temporal knowledge, and (c) event knowledge. Much research has been done to study many different aspects of knowledge, and it would be impossible here to refer to it all. Our modest contribution is from a definition analysis point of view, and was aimed simply at showing what is possible to find in dictionary definitions. The organization we suggest helps understand the type of information in the definitions, and will also help understanding the type of information often seek after in corpora.

3 Moving from Dictionary to Corpora

Now that we have looked in dictionary definitions for inspiration on types of knowledge to be discovered from text corpora, let us move on to discuss the actual activity of discovering Knowledge Rich Contexts (KRCs) in text corpora via the identification of Knowledge Patterns (KPs). To focus our discussion, we will look at one particular corpus on the subject of scuba-diving. The 1M word corpus covers different aspects of scuba-diving, such as medical, trips, equipment, safety, dangers, diving procedures, diving types, etc.[6]

Going back to the mention in the introduction of two main communities having an interest in KRC discovery, that is the Information Extraction (IE) and the Computational Terminology (CT) communities, we can highlight here a first distinction. Within the CT community, such a corpus on scuba-diving is of much interest as it is purposely assembled to contain informative texts (as opposed to narrative [9]). From those texts, terminographers will need to generate definitions for the important terms found, such as cave diving, overhead environment diving, decompression trauma, nitrogen narcosis, and barotrauma. Computational terminologists are interested in terms (concepts) [10,11] and semantic relations that will link these terms and allow them to generate good definitions [12,13,14,15].

In IE, the type of text is usually not specified, but the types of queries and required template filling have typically been of the type "who did what to whom, when, where, how and why", which presupposes narrative types of texts. For example, a text relating a diving accident occurrence could be transposed in a template form. Different systems have been built over the years for such purposes, such as AutoSlog [16], CRYSTAL [17], LIEP [18], PALKA [19], and ESSENCE [20], to name a few.

Relating now to the types of knowledge found in dictionary definitions, Terminological Knowledge Bases (TKB) tend to embody static knowledge, contrarily to IE being more interested in event-related knowledge[7]. The two meet with the dynamic knowledge, more specifically the causal knowledge via the causal agents, of interest in CT, and the how and why questions to be answered in IE.

Static knowledge is a good start for a TKB, and might be sufficient for terminographers to write definitions, but is not sufficient for the TKB to become a domain model. Some researchers [21,22] have started to investigate how KPs could help the discovery of causality, adventuring outside the static type to the dynamic type. Although not expected to be present in a MRD, as causality emerges from the interaction of the different agents within a system, we have argued elsewhere [23] that in fact causality takes root in terminology when we look at causal agents and the function relation. As for events, they are transients and are often not of much interest to store in a knowledge base, unless, these events embody some notion of generality. If

[6] The author would like to thank Elizabeth Marshman for generating the scuba-diving corpus and to Ingrid Meyer for giving us access to it.

[7] Event-related knowledge in its extrinsic form is meant here as opposed to the intrinsic form seen in the definitions of the AHFD in section 2.

they are not "one-time event", or events pertaining to specific characters, if they express how things "normally" are, they do have their place in a domain model.

This illustrates again a main difference between CT and IE. In IE, there is an interest for one-time events, but in CT, there is an interest for generalization. The type of texts will be different, the KRCs of interest will be different, and the KPs will be different. Now, looking at these KPs, let us tackle the issue of generality. This comes as two questions: (1) is a given relation expressed similarly in dictionary and corpora? (2) do corpora domains influence the type of relation to look for beyond the ones identified in the MRD?

3.1 Expression of Semantic Relations – Issues of Generality

Let us look at static and dynamic knowledge, and focus on the first question, the varied expression of a semantic relation. Only one or two KPs have been given for each semantic relation in the Tables of section 2, but the variety is much larger even within the AHFD. In the tradition initiated by Hearst [24], bootstrapping is used as a good way of finding alternate KPs for a relation given known concepts pairs connected through that relation. Our investigation on the scuba-diving corpus has not shown any unusual KPs for the relations of hypernymy and meronymy.

Table 7 shows a small sample of semantic relations with the KRC found including a KP. But a more extensive search, with multiple corpora would be required to do any fair assessment. Meyer [1] had mentioned variability across domains, but most often we are simply concern with variability whether it is within a domain or across. A very interesting work by [25] looks into the notion of variability of dependency structures instead of strings (making it more flexible) and relates this work to work on paraphrasing.

Table 7. Domain-independent information found in scuba-diving corpus

Semantic Relation	Example
Part-of	buoyancy-control, body positioning and propulsion techniques **are part of** both Cavern and Cave Diver training.
Hyperonymy	an air embolism **is another kind of** decompression illness
Cause	a lung over-expansion injury **caused by** holding your breath while you ascend.
Definition	The opening of the eustachian tube **is called** the ostium.
Function	Diazepam **is used to** prevent and treat oxygen convulsions and to control vestibular symptoms.

Now, moving on to the second question given above, Table 8 showing how the individual domain of scuba-diving contains its own domain specific relations forces us to answer yes. In fact, we have argued elsewhere [26] that any list used should depend on the application and that difference in many lists suggested are often issues of granularities, therefore suggesting a hierarchical organizational view of relations.

Table 8. Domain-specific information found in scuba-diving corpus

Semantic Relation	Example
Emergency measure	Pure oxygen **is first aid for** any suspected decompression illness
Symptom	The most common barotrauma **symptom** a diver experiences may be mild discomfort to intense pain in the sinus or middle ear.
Risk prevention	Keeping your SPG and high-pressure hose clipped to your left-hand side significantly reduces **the risk of** gauge damage entanglement.

Although outside the scope of this paper, it would be quite interesting to do an across-domain study to see if resemblance and differences can be merged within a hierarchical schema. Such integration attempts have been performed on concepts, such as in the work of [27] merging terms with the Wordnet [28] hierarchy. But to our knowledge such an across-domain hierarchical integration of semantic relations has never been performed.

4 Knowledge Patterns: Definition and Productivity

If KPs provide our entry point into KRCs of the corpora, whether it is static, dynamic or event knowledge, we must have a flexible way of defining them. We will first present our view of defining KPs as including lexical, syntactic and semantic information. This expands on the more common lexical-syntactic view by including a semantic component. We then look at pattern productivity and discuss their use as entry points in the text.

4.1 Toward Lexical-Syntactic-Semantic Patterns

Lexical patterns found through string matching are the simplest form of KPs. They are useful but quite limited in their capacity to retrieve KRCs. Table 9 shows some examples of different types of knowledge (defined in Section 1) found with lexical patterns.

Table 9. Lexical information in KPs.

KP	Semantic Relation	Knowledge Type
is a kind of	Hyperonymy	Static – Paradigmatic
is a tool to	Function	Static – Usage
is to cause to	Causal	Dynamic-Causal
is a person who	Agent	Event-related
to show that	Reason	Event-related

To introduce flexibility, we move toward syntactic information. This simply assumes a link to a part-of-speech (POS) dictionary and does not necessarily require any complex grammatical analysis. It does not even assume any kind of POS tagging which usually implies a disambiguation step. In our research, POS are used as tokens for pattern definition. The syntactic tokens add a lot of flexibility to the search without the burden of doing actual parsing. Table 10 shows some examples.

Table 10. Syntactic information in KPs.

KP	POS	Relation	Expected variations
is a *|a group of	Adjective	Group	large, small, eclectic
is a tool *|p	Preposition	Function	to, for
is to *|r make	Adverb	Causal	really, largely, principally
is what is *|v	Verb	Result	done, left, gained
is a *|n who	Noun	Agent	person, animal, doctor, student

Some POS provide a much larger search space than others. Nouns, for example which account for much more than fifty percent of the words in a dictionary, will generate a large set of answers. It is therefore important to make sure that such a POS is integrated in a pattern which restricts the set of possible nouns. Last row of Table 10 for example shows a pattern with the relative pronoun “who” which will exclude all objects from the preceding noun. On the other hand, a category such as preposition, since a closed-set category (function words), will be much more limiting.

The same way as the relative pronoun “who” above can be seen as restricting the semantic class of the previous noun, we can imagine that knowing the semantic class of a word can help restrict the possible patterns in which they can occur. The possibility of including semantic information in the KP assumes the availability of some lexical-semantic resource. Let us focus on two paradigmatic relations: hyperonymy and synonymy. The same as the syntactic token "noun" (*|n) could be replaced by any noun in the dictionary, a semantic token &home/^home could be replaced by any synonym/hyponym of home. Such semantic information can be found in a resource such as Wordnet [28]. One of main relation in Wordnet is the synonym relation (through the organization of words in synsets) and the hyperonym relation. Wordnet contains a lot of information about hyperonyms in the general common language. Table 11 shows some examples.

Table 11. Semantic information in KPs.

KP	Semantic Link	Semantic category	Expected variations
is a &home for	Synonymy	Home	house, living-place, roof
are ^colour	Hypernym	Colour	brown, blue, green
is a ^time when	Hypernym	Time	period, moment
is an &amount of	Synonymy	Amount	Quantity

Although each kind of token has a value when used alone Table 12 shows some possible interactions, and this is that type of search that would include lexical, syntactic and semantic information, providing a quite flexible way of getting access to KRCs.

Table 12. Interaction of lexical, syntactic and semantic information in KPs.

Relation	Expected variations
is *|d &amount *|p	is an amount for, is a quantity of, is a number of
*|n are parts *|p *|a ^tree	branches are parts of a tree, needles are parts of a pine tree
*|v with ^instrument	write with pencil, draw with crayon
*|n is a *|a ^animal	elephant is a large animal, canary is a small bird

4.2 Pattern Productivity

Although we can define KPs in a flexible way, we still need to investigate how much these KPs actually are the good guides or good entry points into KRCs. This leads us to the question of evaluation. The problem with evaluation is that it usually implies a task for which anticipated results exist, and against which an automated procedure can be compared. In an IE task, texts can be read manually and questions prepared in relation to the information in the text. For CT, it's not as clear what a good evaluation measure is, since the goal is not really to answer to singled out questions, but to "all" eventual questions by providing definitions for terms. So it is trying to get as much knowledge as possible. Still, researchers in CT wish to evaluate the productivity KPs and sometimes use a measure of noise and silence, the complements of precision and recall traditionally use in IE. The calculations are prone to errors since they require a human manually looking through a corpus of texts, and for a singled out semantic relation, for a singled out KP, evaluate noise and silence. This process, even if informative, is also questionable, as it is totally depended on the corpus used, and therefore makes the measures very subjective. Without going manually through the whole corpus, we can minimally evaluate how noisy a pattern is. We can also evaluate what we will define as *relative productivity* by comparing the number of good KRCs identified by all the patterns used and assuming a uniform distribution across patterns.

Focusing on the function relation, Table 13 presents a list of patterns with their frequency of occurrence, noise and relative productivity. We can note in the second to last pattern the use of a wildcard "design*" to cover: design, designs, designed. The last pattern uses POS tag of preposition "used *|p" allowing the retrieval of 47 patterns distributed as the following coverage: used to (15), used in (9), used by (8), used for (5), used with (4), used as (2), used up (1), used like (1), used on (1), used after (1).

A KP with a relative productivity as good as all other patterns would give 100%. Let us call NbExp the number of expected occurrences given a uniform distribution across patterns, the relative productivity would be calculated as: Number of good occurrences – NbExp / NbExp.

A pattern is noisy if sometimes it is indicative of a semantic relation and sometimes not. A different form of ambiguity is when a single pattern can lead to different possible semantic relations. This is the problem we face when we investigate prepositions as patterns. In Table 14, we show a few prepositions and their ambiguity. We went back to the AHFD for the examples, as they provide very clear and simple examples.

Table 13. Statistics on different KPs for the function relation. (i – Total number of occurrences of pattern, ii – Number of occurrences NOT showing a KRC, iii – Percentage of occurrences NOT indicating "function", iv – relative productivity)

Pattern	(i)	(ii)	(iii)	(iv)	Positive Example	Negative Example
serve to	1	0	0%	-91%	Reef hooks (…) may **serve to** limit damage to both reef and diver.	
Useful for	1	0	0%	-91%	Some chemotherapy is **useful for** marine animal injuries.	
made to	2	0	0%	-82%	small incision needs to be **made to** extract the spine	
Intended for	1	0	0%	-91%	regulator second stage **in-tended for** use as an octopus.	
Design* to	28	2	7%	136%	our lungs are **designed to** breathe gas	They are similar in **design to** the active-addition oxygen rebreathers
used *|p	47	11	23%	227%	drugs like those **used to** control cold symptoms	I **used to** go so far as to tell people ….
Total	80	13	16%			

Table 14. Highly ambiguous KRCs

KRCs	Semantic Relation	Example
for	Function	A carpenter has a box **for** his tools.
	Recipient	I bought this book **for** you.
	Direction	People can reach **for** the sky, but they can't touch it.
	Duration	We played baseball **for** 2 hours.
in	Location	Fish swim **in** the water.
	Point-in-time	Ted's birthday is **in** August.
	Manner	Ants live **in** large groups.
	Containment	The people **in** this story are Lisa and David.
of	Part-of	Branches grow out from the branch **of** a tree.
	Containment	Paul was carrying a pail **of** water.
	Point-in-time	The time is 10 minutes **of** four.
	About	Steve drew a picture **of** his brother.
	Possession	You can see the work **of** many artist in a museum.
	Material	We made a border **of** stone around the garden.
with	Containment	Joe's hamburger can **with** onions on it.
	Part-of	A giraffe is an animal **with** a long neck.
	Instrument	Brian dug a hole **with** a shovel.
	Manner	Attention is looking and listening **with** care.
	Accompaniment	To march **with** someone means to take the same size steps at the same time.

This case is interesting as ambiguous prepositions might be very problematic in a IE system, and unfortunately prepositions are some of the most important indicators of event-knowledge, but it might not be so problematic in a CT system, since the terminographer is looking for any interesting information, and is usually restricting the contexts of search to ones containing a term of interest.

It is important though in an interactive system to reduce the load on the user as much as possible [29]. Some even suggest that the user should never be bother with defining patterns KPs but should just be given KRCs and say yes-no (implying that a pattern learner listens in the background) [30]. Depending on how well the learner performs, this might be more or less burden on the user, if he has to constantly be asked yes/no on inappropriate KRCs.

5 SeRT – Knowledge Acquisition Tool

In this section, we briefly describe SeRT (Semantic Relations in Text). SeRT is a Knowledge Acquisition tool which relies at its core on the hypothesis that explicit knowledge is acquired via the discovery of semantic relations, and these semantic relations are expressed via KPs.

In [31] further details are given about some of SeRT functionalities included in an early version, but as an overview, we can say that SeRT has (1) a Term Extraction module (finding a list of terms in a corpus), (2) a Semantic Relation Search module, which implements in a limited way the lexical-syntactic-semantics search (3) a storage capacity comprising two aspects a) a list of semantic relations and their patterns, b) a list of semantic relations and the concepts they link to generate a knowledge base of the extracted information, (4) a visualization module for the knowledge base. For future work, we will focus on the integration of the Wordnet resource with our search module to augment the semantic aspects. For now, the only semantic relations between concepts known are the ones that are iteratively added to the knowledge base.

SeRT is being developed as an aid to a computational terminologist or a knowledge engineer. In a highly interactive way, it gives the human user a flexible tool to discover Knowledge Rich Contexts (KRCs) through the definition of Knowledge Patterns (KPs). A KRC can be looked at within a small context (window with 4 words on each side) to quickly eliminate invalid sentences. A chosen KRC can then be seen in a large context (two sentences) to be looked at by the human user for identification of concepts and semantic relations to be entered in the knowledge base.

To help a new user get started, a list of semantic relation is available to the user, and for each, a list of knowledge patterns is also available. As the user goes through the knowledge acquisition process, the new patterns and new semantic relations identified are added.

The use of SeRT allows the construction of a Knowledge Base on a specific domain, such as the extracts presented in Section 3, on the topic of Scubadiving.

6 Conclusion

Our purpose in this research was to investigate Knowledge Patterns (KPs), as they are the key entry points to the discovery of Knowledge Rich Contexts (KRCs), contexts containing valuable information. We discuss the notion of information value by looking at the needs of two communities interested in knowledge acquisition, the Information Extraction and the Computational Terminology communities. A small children's dictionary served as the starting point to make the link between KPs and semantic relations to be included in a knowledge base. This also allowed us to regroup the type of knowledge present in the definitions in three categories: static, dynamic and event-related, and to make an attempt at characterizing which type is of more value for each community. We further looked at text corpora where further domain-specific relations will need to be defined as well. We provided some ideas on the definition and the productivity evaluation of the KPs. The definitional framework we suggest combines lexical, syntactic and semantic information. The evaluation framework we suggest limits the human burden by considering only the KRCs instead of all the sentences in a corpus, and gives a relative notion of productivity. Finally we have presented SeRT which is an interactive tool for knowledge acquisition. Further investigations, as mentioned before will include integration of the semantic capabilities of SeRT with Wordnet, as well as further refinement to pattern definitions, to include even more complex patterns, such as (as NP1 vb1, NP2 vb2) && (vb1 antonym vb2) [32] which includes logical operators. This will render SeRT an even more flexible and powerful tool for helping humans in their process of knowledge acquisition and organization.

References

1. Riloff, E.: An empirical study of automated dictionary construction for information extraction in three domains. In: Artificial Intelligence 85, (1996) 101-134
2. Meyer, I.: Extracting Knowledge-rich Contexts for Terminography: A Conceptual and Methodological Framework. In: Bourigault, D., Jacquemin, C., and L'Homme M.C. (eds): Recent Advances in Computational Terminology, John Benjamins, (2001) 279-302
3. Chodorow, M.S., Byrd R.J., and Heidorn, G.: Extracting semantic hierarchies from a large on-line dictionary. In: 23rd Annual Meeting of the Association for Computational Linguistics, (1985) 299-304
4. Ahlswede, T., and Evens, M: Generating a relational lexicon from a machine-readable dictionary. International Journal of Lexicography 1(3), (1988) 214-237
5. Dolan, W., Vanderwende L, Richardson, S.D.: Automatically deriving structured knowledge bases from on-line dictionaries. In: The First Conference of the Pacific Association for Computational Linguistics, Vancouver, (1993) 5-14
6. Wilks, Y., Fass, D., Guo, C.-M., McDonald, J., Plate, T., and Slator, B.: Providing machine tractable dictionary tools. In: Pustejovsky, J. (ed.): Semantics and the lexicon, chapter 16, (1993) 341-401
7. Barrière, C., and Popowich, F.: Building a Noun Taxonomy from a Children's Dictionary. In: Gellarstam M. et al. (eds): Proceedings of Euralex'96, Gotebord, (1996) 27-34

8. Fillmore, C.: The Case for Case, In: Bach, E., and Harms, R. (eds): Universals in Linguistic Theory, New York, Holt, Rinehart and Wilson (1968)
9. Kintsch, W. and van Dijk, T.A.: Toward a model of text comprehension and production, Psychological Review, 85(5), (1978) 363-394
10. Kageura, K., and Umino, B.: Methods of Automatic term Recognition: A Review. Terminology 3(2), (1996) 259-290
11. Cabré Castellvi, M.T., Bagot, R.E. and Palatresi, J.V.: Automatic term detection: A review of current systems. In: Bourigault, D., Jacquemin, C., and L'Homme M.C. (eds): Recent Advances in Computational Terminology, John Benjamins, (2001) 53-87
12. Biebow, B. and Szulman, S.: TERMINAE: A linguistics-based tool for the building of a domain ontology. In: 11th European Workshop on Knowledge Acquisition, Modeling and Management (EKAW'99), Dagstuhl Castle, Germany, (1999) 49-66
13. Bowden, P.R., Halstead, P., and Rose T.G.: Extracting Conceptual Knowledge from Text Using Explicit Relation markers. In: Shadbolt, N, O'Hara K., and Schreiber G. (eds): Proceedings of the 9th European Knowledge Acquisition Workshop, EKAW'96, Nottingham, United Kingdom, (1996) 147-162
14. Condamines, A. and Rebeyrolle, J.: CTKB: A Corpus-based Approach to a Terminological Knowledge Base. In: Bourigault, D., Jacquemin, C., and L'Homme, M-C. (eds): Computerm'98, (1998) 29-35
15. Meyer, I., Mackintosh, K., Barrière, C., and Morgan T.: Conceptual sampling for terminographical corpus analysis. In: Terminology and Knowledge Engineering, TKE'99, Innsbruck, Austria, (1999) 256-267
16. Riloff, E.: Automatically constructing a dictionary for information extraction tasks. In: Proceedings of the Eleventh National Conference on Artificial Intelligence, (1993) 811-816
17. Soderland, S., Fisher, D., Aseltine, J., Lehnert, W.: CRYSTAL: Inducing a conceptual dictionary. In: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (1995) 1314-1319
18. Huffman, S.B.: Learning information extraction patterns from examples. In: Lecture Notes in Computer Science, Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing, (1996) 246-260
19. Kim, J., and Moldovan, D.: Acquisition of linguistic patterns for knowledge-based information extraction. In: IEEE Transactions on Knowledge and Data Engineering, 7(5), (1995) 713-724
20. Català, N., Castell, N., and Martin, M.: Essence: A portable methodology for acquiring information extraction patterns. In: ECAI'2000, Proceedings of 14th European Conference on Artificial Intelligence, (2000) 411-415
21. Garcia, D. : Structuration du lexique de la causalité et réalisation d'un outil d'aide au repérage de l'action dans les textes, Actes des deuxièemes rencontres - Terminologie et Intelligence Artificielle, TIA'97, (1997) 7-26
22. Barrière, C.: Investigating the Causal Relation in Informative Texts, Terminology 7(2) (2001) 135-154
23. Barrière, C., and Hermet, M.: Causality taking root in Terminology. In: Proceedings of Terminology and Knowledge Engineering, TKE'2002, (2002)
24. Hearst, M.: Automatic Acquisition of Hyponyms from Large Text Corpora. In: Actes de Coling'92, Nantes. (1992) 539-545
25. Lin, D. and Patel, P.: Discovery of Inference Rules for Question Answering. In: Natural Language Engineering, 7(4). (2001) 343-360

26. Barrière, C.: Hierarchical Refinement and Representation of the Causal Relation, Terminology 8(1), (2002) 91-111
27. Alfonseca, E., and Manandhar, S.: Extending a Lexical Ontology by a Combination of Distributional Semantics Signatures. In: EKAW 2002, LNAI 2473, Springer-Verlag. (2002) 1-7
28. Miller, G.A.: WordNet: a lexical database for English, Communications of the ACM, 38(11), (1995) 39-41
29. Gil, Y. and Ratnakar, V.: IKRAFT: Interactive Knowledge Representation and Acquisition from Text. In: EKAW 2002, LNAI 2473, Springer-Verlag. (2002) 27-36
30. Brewster, C., Ciravegna, F., Wilks, Y.: User-Centred Ontology Learning for Knowledge Management, NLDB 2002, LNCS 2553, Springer-Verlag Berlin Heidelberg (2002) 203-207
31. Barrière, C., and Copeck T.: Building a domain model from specialized texts. In: Proceedings of Terminologie et Intelligence Artificielle, TIA'2001, Nancy, France (2001) 109-118
32. Moldovan, D.I. and Gîrju, R.C.: An interactive tool for the rapid development of knowledge bases. In: International Journal on Artificial Intelligence Tools, 10(1,2) World Scientific Publishing Company (2001) 65-86

Intrinsic Representation: Bootstrapping Symbols from Experience

Stephen David Larson

MIT Computer Science and Artificial Intelligence Laboratory (CSAIL)
200 Technology Square, Cambridge, MA 02139, USA,
stelar@alum.mit.edu

Abstract. If we are to understand human-level intelligence, we need to understand how meanings can be learned without explicit instruction. I take a step toward that understanding by showing how symbols can emerge from a system that looks for regularity in the experiences of its visual and proprioceptive sensory systems. More specifically, the implemented system builds descriptions up from low-level perceptual information and, without supervision, discovers regularities in that information. Then, the system, with supervision, associates the regularity with symbolic tags. Experiments conducted with the implementation shows that it successfully learns symbols corresponding to blocks in a simple 2D blocks world, and learns to associate the position of its eye with the position of its arm.

In the course of this work, I propose a model of an adaptive knowledge representation scheme that is intrinsic to the model and not parasitic on meanings captured in some external system, such as the head of a human investigator.

1 Introduction

In 1998, a group of MIT AI Lab faculty came together to form The Human Intelligence Enterprise, a group committed to the pursuit of understanding human intelligence from a computational perspective. From this group, the Bridge Project was initiated to investigate an interface between language representation and visual representation. The Genesis group, the name of the student group spawned by the Human Intelligence Enterprise to work on the Bridge Project, took an approach to investigating human intelligence that is very timely: to investigate the processes and representations that contribute to *general* human-level intelligence.

Inspired by this pursuit, I have asked the following questions: 1) How can we design representations for intelligent systems that bridge the gap between symbols and their subsymbolic referents, also known as the Symbol Grounding Problem? [Harnad 1990] 2) How can we design representations that are not limited to learning only within specific domains? This paper takes a step toward answering those questions.

Taking inspirations from the cerebral cortex and drawing from some older technologies (Kohonen's self-organizing maps, clustering algorithms, Hebbian association), I have fashioned a model that puts these elements together in a new way. The model is a self-organizing representation capable of representing the statistically salient features of an information space. It can be used to train a system to associate the movement of its arm with the movement of its eye, as well as to associate colored blocks with linguistic symbols representing utterances.

2 Related Work

[Agre and Chapman 1987] investigated the instantiation of general symbols using indexical-functional aspects. Aspects were intended to provide a meaning for objects in a world relative to their usefulness for an agent. [Drescher 1991] looked at how a system can build knowledge on top of knowledge through interaction in a simple micro-world with a hand, an eye, and objects. The system in [Beule, Looveren, and Zuidema 2002] is given information about objects such as their positions and their constituent properties and returns syntactic structures describing notable objects in a world such as "the red square moving to the right". [Roy et al. 2003] describes a physically instantiated robot arm that can pick up objects by associating sensorimotor primitives together.

What the model of Intrinsic Representation shares with the previous works is the desire to associate symbols with subsymbolic descriptions. The following points outline the ways in which the model differs from the approach of traditional symbol systems. The same points explain the features of the model not shared by the previous works.

Symbols' descriptions are discovered from the statistical processing of experience. In almost every traditional symbol system, the symbols *and* their descriptions must be provided before-hand. In the model of Intrinsic Representation, the descriptions of symbols are discovered by processing sensory data in an unsupervised manner.

Symbols' descriptions are equivalent to statistical regularities found in information spaces. The nature of the symbols in traditional symbol systems are as tags whose descriptions are provided by a human designer. Descriptions formed from statistical regularities do not have to be provided by a human designer. As a result, symbols will represent certain classes of things in the world not because a designer finds them useful, but because those things are the most statistically salient in the given information space.

Symbols' descriptions carry their context with them by being situated in information spaces. Traditional symbol systems require context as an extra parameter to make sense of a symbol. In such systems, symbols may mean different things in different contexts. This is because a symbol is thought to be a separable part of a system rather than an intrinsic part of a system. Because symbols within an Intrinsic Representation are derived from information spaces they are inextricably linked to them and carry no separate meaning. For example, a cluster in an information space of visual information cannot be transported into an information space of any other kind of information, its data would not make sense in a different context. As a result, you cannot have a symbol that is not bound to its context.

3 Moving Forward

Representation has been at the center of the field of Artificial Intelligence from its inception. Formal symbolic systems capitalized on the ability to store information about the world in computer memory. The great successes of this approach, chess-playing computers and expert systems to name a few, relied on AI researchers' ability to invent ways of cleverly modeling the outside world.

On the other side of the tracks, the parallel distributed processing approach paved the way for a different approach to representation [Rumelhart and McClelland 1986]. Rather than symbols, the values stored in the hidden nodes of pattern-analyzing networks provided a new perspective on what a representation could be. A great emphasis was placed on the learning of patterns. The ability for a network to produce the appropriate input-output behavior was taken as the proof that the representation formed by the hidden nodes was appropriate.

The rise of nouveau AI and the pursuit of insect-like intelligent robots ushered in an era that sought to dismiss internal representations, opting instead for using the world as its own best representation [Brooks 1991]. The early success of this approach demonstrated a third perspective of representation that turned the classical view on its head.

The proper recognition of representation as the cornerstone of Artificial Intelligence helps put the road ahead into greater focus. As we understand more about the brain we gain an increased appreciation for the hard problem of representation that evolution has had to solve. This is the problem of the appropriate organization of information, a problem common to both organisms and man-made intelligent systems. While some might be hesitant to label the recognition of stimuli by simple animals as representation (as Brooks might), I would respond that there is much profit to be gained from broadening the definition. Doing so immediately leads us to the conclusion that simple organisms provide us with "page one" of the evolutionary history of representation. Our simpler evolutionary ancestors did not have the luxury of high-level concepts as humans do to help them craft clever taxonomies. Their only view of the world were the low-level patterns of activation they received from arrays of sensor cells. How can we model the processes that enabled them to distinguish food from poison and mate from attacker? For such simple creatures, we must boil the definition of representation down to its most basic and pure: representation is about grouping similar things together and placing dissimilar things farther apart.

From this viewpoint, it becomes clear that the two strategies of constructing representations, man-made and evolutionary, differ in an important way. The groupings of man-made representations are based on human choice of what is similar and what is dissimilar. The sensory groupings created by the nervous systems of simple organisms are certainly not. But what, if not the design of a creator, could be responsible for the appropriate groupings of stimuli that simple organisms exhibit?

In order to answer that question, we are forced to speak the language of the neuron– statistics and information theory. Biological systems have some help in making sense of the world from its sensory information: the world is not filled with Gaussian white noise. Natural environments contain patterns that are identifiable after the right processing has been carried out.

The information that an organism receives from its sensor cells can be thought of as forming a high-dimensional vector space of information, or "information space" [Larson 2003]. For each organism, it is most advantageous to be able to process and react to sensory data within the those subspaces most relevant to survival. Seen from this angle we might explain the emergence of biological representation as the result of the natural selection of those organisms best able to model those subspaces.

If this is indeed the strategy that Mother Nature has exploited to enable simple organisms with the ability to represent the world, could we follow in her footsteps? Could a representation be built up from low-level sensory information by making statistical distinctions between similar and dissimilar input? Could such a representation self-organize from the statistical regularity of its input the ability to tell the difference between sensory patterns rather than relying on a magic "if" statement which, in turn, relies on human-designed assumptions of similarity? Such a representation is the goal of the model of Intrinsic Representation.

4 The Model

Figure 1 is the key diagram for understanding the model of Intrinsic Representation. Two information spaces, i and j, are shown side by side. At the bottom of the diagram, sensor arrays receive streams of data from the outside world. Such arrays could be imagined as a patch of skin or light-sensitive cells in the retina. As data comes into the system through these sensors, they travel to a subsystem devoted to organizing and storing the regularities in the data. This subsystem arranges these regularities with respect to their similarity, placing similar regularities in proximity, and dissimilar regularities farther apart. After a critical period, the regularities are grouped into clusters of high similarity. Once grouped, a cluster gains the ability to act as a unit that can be activated and deactivated. Data in the incoming stream can be treated as the trigger for the activation of the cluster of which they are a member. As clusters are activated by incoming data, they are associated together by their frequency of coincidence. The more often two clusters are active simultaneously, the more associated they

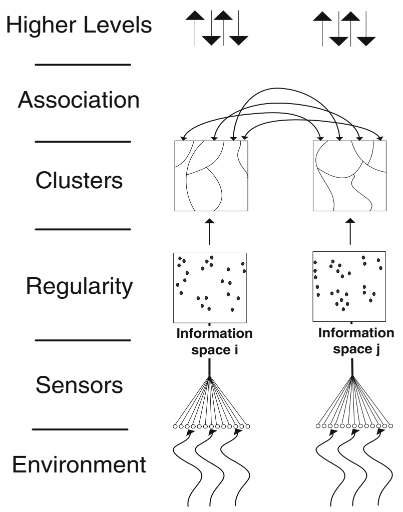


Fig. 1. The model of intrinsic representation.

will become. The resulting trained system treats incoming patterns as members of a class, and can react to the activation of that class.

4.1 Architecture and Implementation

This section discusses the architecture and the implementation of a computer program demonstrating the model of Intrinsic Representation.

The architecture of the program is divided into four major areas:

1. A Blocks World Simulation
2. A Self Organizing Map
3. A Cluster Set
4. Cluster Associations

Each of these blocks feeds into the next in succession. Data flows into the system from a simple 2D blocks world that provides an environment for the present system. In the blocks world, there is an arm with a simple grip useful for grasping objects, much like the arcade novelty that invites players to catch prizes by controlling a robot arm. There is also an eye, whose focus is represented by a square that moves around the environment. The eye and the arm are the sources of sensory and motor interaction with the world. There are also blocks in the world that can be picked up and stacked on top of one another. Simple physics modeling enables such constraints as gravity and friction.

Both the eye and the arm are equipped with sensors. Every sensor is normalized, and thus reads a real value between zero and one. The eye has retina sensors arranged in a 2D array that register the red-green-blue value of any spot they are over. The eye also has sensors that report its horizontal and vertical orientation, mimicking the information the brain receives from the muscles that position the eyes.

The arm has proprioceptive sensors and tactile sensors on its grip. The proprioceptive sensors report how far it is extended both horizontally and vertically, mimicking feedback from muscles and joints. The tactile sensors report collisions with objects. A vector of sensor values is read from the eye and arm each time their values change (see figure 2).

As data are output from a given device, they are received by a self-organizing map [Kohonen 2001]. For the experiments conducted, two self-organizing maps are used; one for the eye and one for the arm. The map is specialized to its input only in the respect that its dimensionality must match. As vectors are received, the map's self-organizing process iterates following a growing Kohonen map algorithm [Dittenbach, Merkl, and Rauber 2000].

The maps undergo a self-organizing process that is driven by the visual or proprioceptive information that they are set to receive. The maps are allowed to self-organize and are later prevented from acquiring new data (see figure 3). At this point, a clustering algorithm is used on the data held by the map to separate the major clusters in the space. This is visually represented as finding the discrete sets on the map which correspond to the groupings of the most similar cells (see figure 4).

Once clusters have been established, each of the clusters on the map is given its own unique identifier, and a new data object is created to represent it. This data object keeps track of the similarity measure of the cells inside the cluster, as well as the indices of the cells it contains.

This data object also keeps track of any associations that a cluster has made with other clusters, even those outside of the current map. In this sense, the cluster acts like a single unit whose activation can become associated with the activation of other units. The association process is a simplified version of Hebbian learning. When clusters are activated by sensory data simultaneously, a counter labeled with the names of those clusters is incremented. Clusters are active when input matches most closely with a cell within the cluster. The cluster association system identifies the clusters most associated with any particular cluster. Clusters are never associated with clusters from the same map.

Once the system has been trained, the associations can be used to activate units between modalities. This allows it to exhibit behavior that neither modality separately would be able to accomplish as easily by itself.

4.2 Three Levels of Representations

The architecture I have just described takes advantage of three different levels of representation arranged in a hierarchy. The following describes these representations in greater detail.

Map Cells. A map cell is a single cell in a self-organizing map. Each cell keeps track of a subset of information that enters from the environment during the initial self-organizing process. Each cell in the map stores an n-dimensional vector. The data contained in that vector corresponds to a point in an n-dimensional information space. In the context of a self-organizing map, a map cell represents a point in an n-dimensional information space that is representative of a class of points that the map has observed.

Clusters. The cluster is one representational level higher than a map cell. It is a collection of statistically similar map cells in a self-organizing map. Because cells represent vectors in an n-dimensional information space, a cluster therefore is a collection of statistically similar vectors.

In the context of a self-organizing map, a cluster represents a collection of cells that have been arranged near each other by the SOM process [Kohonen 2001]. The SOM process also arranges cells such that dissimilar cells are distant in the space. As a result, the cells inside a cluster are similar to each other, but different from other major clusters in the map.

Putting this together with what we know about a map cell, a cluster represents a subspace of an n-dimensional information space by storing a collection of n-dimensional vectors. While these vectors are not a basis for that subspace as in linear algebra, they are a collection of statistically interesting vectors that are representative of a class of points in that subspace.

Associations/Symbols. Associations are a representational level higher than clusters. In general, they represent a collection of clusters, usually two. They are discovered between different maps through the process of Hebbian learning. This creates connections between clusters that are active simultaneously in different maps.

5 Experiments

Learning About Blocks. In this section I demonstrate how the system learns about blocks in a simple 2D blocks world at all three of the representational levels just described. In a nutshell, the system discovers implicit representations of blocks in its world in an unsupervised manner. Then it assigns explicit symbols to those implicit representations in a supervised manner.

The first experiment involves two self-organizing maps that read data from a blocks world. One map reads data from the eye in the blocks world, and the other map reads data from the arm. Figure 2 illustrates the features that make up the input data vectors.

Additionally, there is a third fixed map that contains linguistic symbols that both the eye map and the arm map are allowed to associate with. This linguistic symbols map is artificially manipulated by the experiment to act as supervisor. When the eye or the arm come in contact with a block of a particular color, this linguistic symbols map will activate a cluster that corresponds to the appropriate symbol, in an attempt to model a distinguishable utterance identifying the block. The task before the learner, then, is to build up a representation that matches the correct utterances with the things it sees and feels in the world.

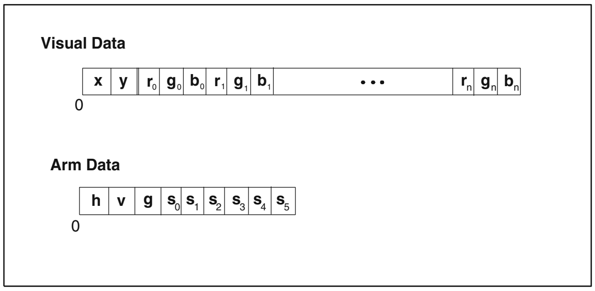


Fig. 2. A guide to the features stored in the visual and arm data vectors that arrive at the eye map and the arm map respectively. The visual data includes x and y coordinates for the eye position, followed by r, g, and b which store the red, green and blue values for each pixel position in the eye. The arm data includes h, v, and g which store the horizontal and vertical components, the width of the grip, and 6 sensor values that are read from sensors on the gripper/hand.

Training The Maps. First, the eye map is trained on the objects in the scene separately. To accomplish this, the eye is shifted between the objects in the scene several times. The scene consists of a red, a green, and a blue block, plus the arm. While viewing a single object, the eye is allowed to scan by taking a circular path around it. As the eye moves from object to object, the eye map grows to capture more of the views it is receiving. Figure 3 illustrates the progress of the eye map in this phase of training. The result of this process is a map that captures views of the objects in the scene.

Once the eye map has been trained with the stationary objects, the arm is trained on the blocks in the scene by being moved to each block in succession, grasping it, picking it up, moving it around, lifting it up, and eventually dropping it. The result of this process is shown in figure 5 and illustrates a distinction between the arm with something in its grasp, and the arm without something in its grasp.

While the arm is being trained, the eye is focused on the actions of the arm, thus updating the eye map to include the interactions between the arm and the blocks. Figure 4 illustrates the resulting map.

Clusters. Following the training of the maps, Figure 4 shows the eye map segmented into clusters, and figure 5 shows the arm map segmented into clusters.

Associations. Once the maps have been segmented into clusters, their clusters are associated together. The training process is a repetition of the arm training—the arm travels to each block, grips it, picks it up, and moves it around, while the eye looks on.

During *this* training phase, both maps are given the opportunity to form associations with the special utterance map, which contains symbols that are activated or deactivated during training. When the eye and the arm are interacting with a block, the appropriate symbol is activated. When the eye and

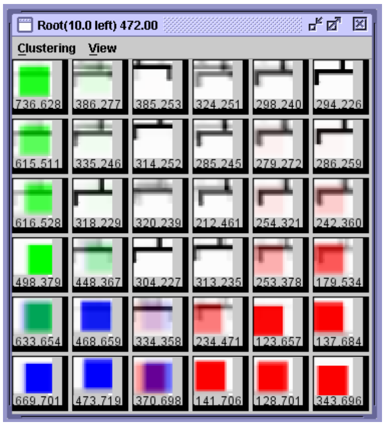


Fig. 3. A visual representation of the eye map partially trained on the four objects in the simple 2D blocks world. Each cell is a bitmap representation of the rgb values of underlying visual input data vectors. In the bottom left, the map shows a few views of the blue block. In the bottom right several cells show views of the red block. In the upper right, middle and center, several views of the gripper/hand can be seen. The cells along the upper and middle left show views of the green block. The "smeared" cells are the product of different inputs that came soon after one another and thus created cells that are on a cluster border.

the arm leave the area of the blocks, no symbols are asserted. The symbols are `(Red_Block)`, `(Blue_Block)`, and `(Green_Block)`. These symbols are undivided; while to us they imply a combination between a color and a shape to a human reader, to the system, they are only treated as distinguishable atomic tags. This is consistent with the notion that symbols by themselves are just tags until associated with sub-symbolic meaning.

The left half of table 1 shows the results of the association between these symbols and the clusters in the eye map. Higher frequency corresponds to greater association between the symbol and the cluster. The cluster IDs are the same as the cluster IDs in figure 4 to provide the reader a sense of the clusters that the symbols are being associated with. The right half of table 1 shows corresponding results for association between the symbols and the clusters in the arm map. These cluster IDs correspond to the ones in figure 5.

Generally, the associations tend to fall into the largest clusters that, to the eye, appear to be the most representative of the block in the symbol. In this case, however, table 1 shows that the symbol `(Blue_Block)` is instead most associated with a smaller cluster, though the largest cluster corresponding to the symbol `(Blue_Block)` appears farther down on its list. This demonstrates how even small clusters can represent important parts of the information space.

One of the interesting results of table 1 is that the `(Blue_Block)` symbol and the `(Green_Block)` symbol are both most highly associated with cluster 166. This cluster corresponds to a state where the arm is gripping something. The

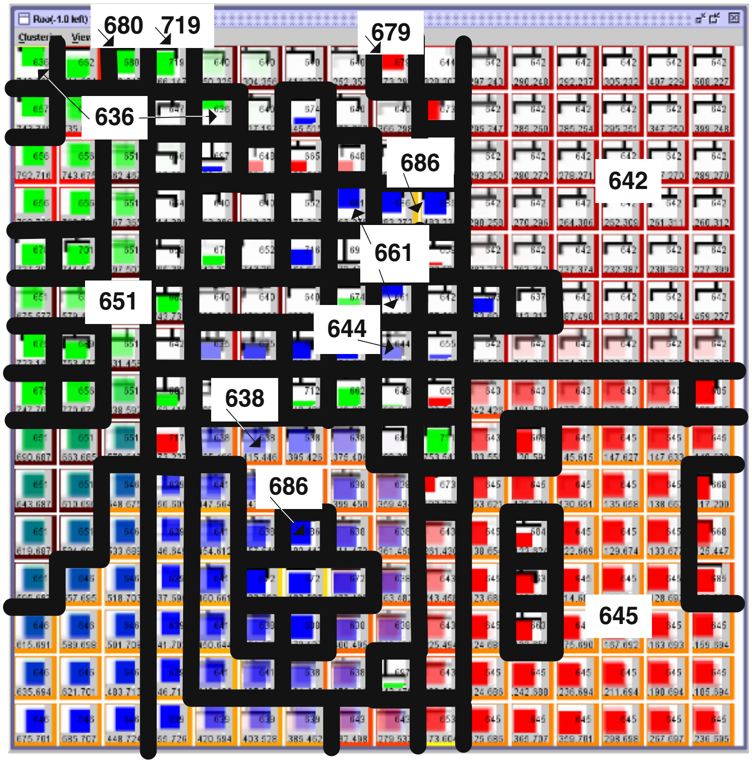


Fig. 4. A trained eye map with clusters overlaid. Here, the images visually represent the values that the arm sensors can take on. The cluster IDs of relevant clusters are highlighted.

Table 1. The major associations between symbols and clusters in the eye map and the arm map

SYMBOL	Eye Map Cluster ID	Frequency	Arm Map Cluster ID	Frequency
`(Red_Block)`	645	69	173	78
			169	35
`(Blue_Block)`	686	35	166	63
	661	24	165	22
	638	10	170	14
`(Green_Block)`	651	80	166	100
	680	26	168	10
	636	23		
	719	20		

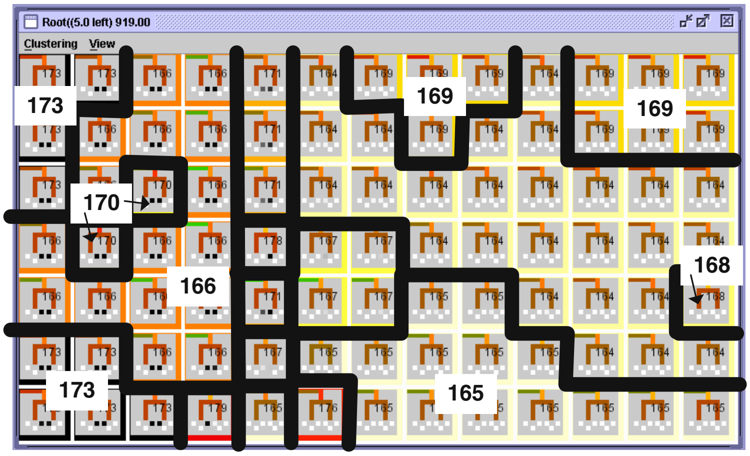


Fig. 5. A trained arm map with clusters overlaid. The cluster IDs of relevant clusters are highlighted.

associations further down the list, which match to different clusters, dissociate the two symbols so that they can be independently identified.

This experiment demonstrates the ability of the system to ground symbols in sensory data. The system has acquired sensory data and organized them into classes in an unsupervised fashion. Later, this data was associated with symbols in a supervised fashion. Thus, the symbolic information stored in the special utterance map has been grounded by the subsymbolic data from the environment.

Associating Hand with Eye. In this section I demonstrate how the system can use associations formed between the position of the eye and the position of the arm to enable the arm to move to the current position of the eye.

The training of the eye map and the arm map follows a simpler process than in the previous blocks-learning situation. There is no third map whose activity is managed by the experimenter. In this experiment, the two maps are allowed to associate directly with one another. Training proceeds by fixing the position of the eye to that of the gripper/hand, and moving them together to random points in the space. The space does not have any blocks into it.

What we expect from this training is for the eye map to have stored very similar images since it views the same part of the gripper for all time. Because of this, the x and y components of the eye data should become more significant. The resulting eye map should have an even distribution of x and y positions across it. The arm map will emphasize the horizontal and vertical components as they are the only ones that are varied.

Figure 6 shows the trained eye map after the clustering process has run. Notable clusters are highlighted. Our expectation of a smooth and continuous map of x and y values has been met. Figure 7 shows the trained arm map after

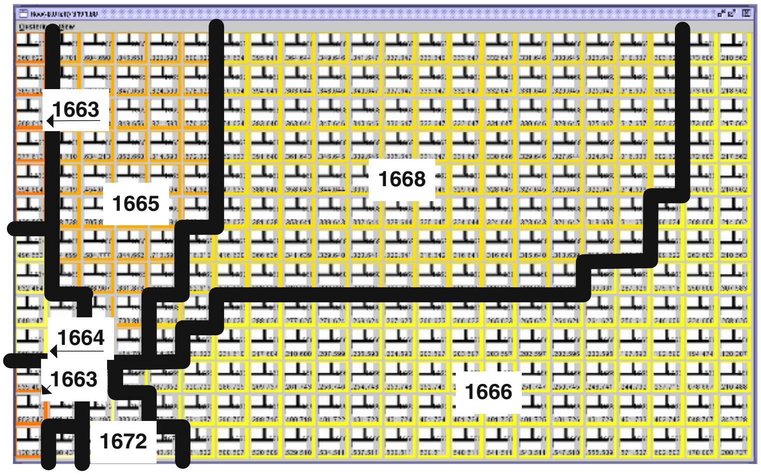


Fig. 6. A trained eye map with clusters overlaid. Relevant clusters are highlighted. All the cells show closely similar views of the world, but the x and y components vary across the map.

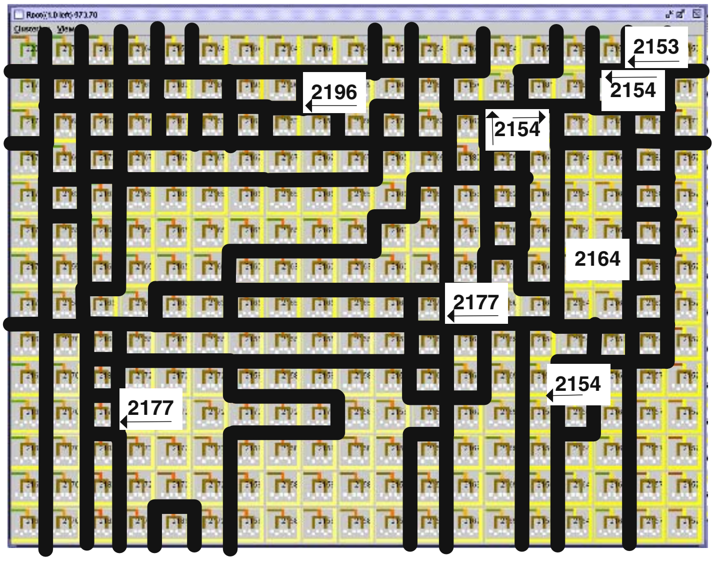


Fig. 7. A trained arm map with clusters overlaid. Relevant clusters are highlighted. While the gripper sensors and grasp remain constant, the map self-organizes using the horizontal and vertical components of the data alone.

the clustering process has run, with notable clusters highlighted. Table 2 shows the data from the cluster associations.

Table 2. Associations between the eye map and the arm map.

Eye Map Cluster ID	Arm Map Cluster ID	Frequency	Eye Map Cluster ID	Arm Map Cluster ID	Frequency
1663	2177	152	1663	2186	49
1663	2154	140	1663	2176	45
1663	2164	136	1663	2182	45
1663	2153	127	1664	2153	27
1663	2181	125	1664	2181	24
1663	2196	125	1664	2152	20
1663	2169	101	1664	2187	20
1663	2152	92	1665	2196	20
1663	2156	81	1665	2164	19
1663	2155	78	1665	2181	15
1663	2166	72	1665	2153	15
1663	2187	63	1666	2177	36
1663	2163	58	1666	2154	33
1663	2180	57	1666	2181	29
1663	2167	56	1666	2180	19
1663	2157	55	1672	2196	17
1663	2165	52			

These associations allow the arm to move into a region near the eye's view. To accomplish this, the currently active cluster in the eye map is identified using the current state of the eye (see figure 8). Then, the arm cluster most highly associated with the currently active eye cluster is selected from the arm map. The vectors from the cells within this cluster are averaged together, and the arm is driven toward this average vector. A feedback process is used whereby the arm continues to move in the direction of the average vector so long as its current state vector does not match it. Note that while this experiment focused on moving the arm to the eye, the inverse could also be accomplished with the same trained maps by simply reversing the process just described.

This experiment demonstrates a deeper use of the model of Intrinsic Representation; to enable action. Through an unsupervised process, the different coordinate systems of the eye and the arm are trained to be compatible, and to allow one to "speak the same language" as the other.

This experiment also shows how exchanges can be made between sensory systems without involving any linguistic symbols whatsoever. This observation expands what we mean by a symbol. The experiment demonstrates the equivalence of 1) associations made with clusters that store linguistic symbols and 2) associations made with clusters that store other kinds of sensory data. Of course, in the brain, linguistic symbols *are* just another kind of sensory data, so this is just as it should be.

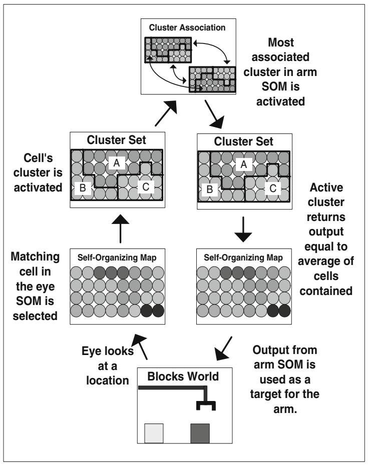

Cluster Association
Most associated cluster in arm SOM is activated
Cluster Set
Cluster Set
Cell's cluster is activated
A
B
C
A
B
C
Active cluster returns output equal to average of cells contained
Matching cell in the eye SOM is selected
Self-Organizing Map
Self-Organizing Map
Eye looks at a location
Blocks World
Output from arm SOM is used as a target for the arm.

Fig. 8. A schematic showing how a trained representation can give rise to behavior. The self-organizing map and cluster set on the left store sensory information from the eye, while on the right they store sensory information from the arm. Starting at the blocks world at the bottom, we see that when the eye looks at a location, that information matches a cell in the eye map. This activates a cluster of cells in the eye's cluster set. The eye cluster set has been associated with the arm cluster set, and thus, a cluster in the arm map is activated. The average value of the cells in that cluster is outputted to the arm, causing it to move to a location.

6 Contributions

In this paper, I have:

1. Outlined the differences between Intrinsic Representation and other approaches to the Symbol Grounding Problem.
2. Justified the need to pursue self-organizing representations as an attempt to recapitulate the evolution of biological representations.
3. Elucidated a hierarchical, self-organizing representational system called Intrinsic Representation that acquires symbols in an unsupervised manner by extracting statistically salient information through interaction with its environment.
4. Described the architecture of Intrinsic Representation through its three key levels of representation: the map cell, the cluster, and the association.
5. Provided the results of two experiments carried out using this representational system instantiated in a computer program.

References

[Agre and Chapman 1987] Agre, P.E., and D. Chapman. 1987. "Pengi: An Implementation of a Theory of Activity". Edited by Morgan Kaufmann, *Proc. Sixth National ConferenceAmerican Association for Artificial Intelligence*, All ACM Conferences. American Association for Artificial Intelligence, 268–272.

[Beule, Looveren, and Zuidema 2002] Beule, Joachim De, Joris Van Looveren, and Willem Zuidema. 2002. "Grounding Formal Syntax in an Almost Real World". Ai-memo 02-03, Vrije Universiteit Brussel, Artificial Intelligence Laboratory.

[Bickhard and Terveen 1995] Bickhard, Mark H., and Loren Terveen, eds. 1995. *Foundational Issues in Artificial Intelligence and Cognitive Science.* Advances In Psychology no. 109. North-Holland.

[Brooks 1991] Brooks, Rodney A. 1991. "Intelligence without representation". *Artificial Intelligence*, no. 47:139–159.

[Dittenbach, Merkl, and Rauber 2000] Dittenbach, M., D. Merkl, and A. Rauber. 2000, July 24. – 27. "The Growing Hierarchical Self-Organizing Map". Edited by S. Amari, C. L. Giles, M. Gori, and V. Puri, *Proc of the International Joint Conference on Neural Networks (IJCNN 2000)*, Volume VI. Como, Italy: IEEE Computer Society, 15 – 19.

[Drescher 1991] Drescher, Gary L. 1991. *Made-up Minds.* Cambridge, Massachusetts: MIT Press.

[Fuster 2003] Fuster, Joaquin M. 2003. *Cortex and Mind: Unifying Cognition.* Oxford University Press.

[Harnad 1990] Harnad, Stephen. 1990. "The Symbol Grounding Problem". *Physica D* 42:335–346.

[Kohonen 2001] Kohonen, Teuvo. 2001. *Self-Organizing Maps.* Third. Springer Series in Information Science no. 30. Springer.

[Larson 2003] Larson, Stephen David. 2003, September. "Intrinsic Representation: Bootstrapping Symbols From Experience". Master's thesis, Massachusetts Institute of Technology.

[Minsky 1988] Minsky, Marvin. 1988. *The Society of Mind.* New York: Simon and Schuster.

[Roy et al. 2003] Roy, Deb, Kai-Yuh Hsiao, Nikolaos Mavridis, and Peter Gorniak. 2003. "Ripley, Hand Me The Cup! (Sensorimotor representations for grounding word meaning)". *Int. Conf. of Automatic Speech Recognition and Understanding.*

[Rumelhart and McClelland 1986] Rumelhart, David E., and James L. McClelland. 1986. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* Volume 1. MIT Press.

[Winograd 1971] Winograd, Terry. 1971, Feb. "Procedures as a Representation for Data in a Computer Program for Understanding Natural Language". Mit ai technical report 235, MIT.

[Winston 1993] Winston, Patrick. H. 1993. *Artificial Intelligence.* Third. Reading, Massachusetts: Addison-Wesley.

Sequential Consolidation of Learned Task Knowledge

Daniel L. Silver and Ryan Poirier

Intelligent Information Technology Research Laboratory,
Jodrey School of Computer Science, Acadia University,
Wolfville, Nova Scotia, Canada B4P 2R6
danny.silver@acadiau.ca

Abstract. A fundamental problem of life-long machine learning is how to consolidate the knowledge of a learned task within a long-term memory structure (domain knowledge) without the loss of prior knowledge. Consolidated domain knowledge makes more efficient use of memory and can be used for more efficient and effective transfer of knowledge when learning future tasks. Relevant background material on knowledge based inductive learning and the transfer of task knowledge using multiple task learning (MTL) neural networks is reviewed. A theory of task knowledge consolidation is presented that uses a large MTL network as the long-term memory structure and task rehearsal to overcome the stability-plasticity problem and the loss of prior knowledge. The theory is tested on a synthetic domain of diverse tasks and it is shown that, under the proper conditions, task knowledge can be sequentially consolidated within an MTL network without loss of prior knowledge. In fact, a steady increase in the accuracy of consolidated domain knowledge is observed.

1 Introduction

The majority of machine learning research has focused on the single task learning approach where an hypothesis for a single task is induced from a set of training examples with no regard to previous learning or to the retention of task knowledge for future learning. In contrast, humans take advantage of previous learning by retaining task knowledge and transferring this knowledge when learning a new and related task. Life-long learning is a relatively new area of machine learning research concerned with the persistent and cumulative nature of learning [21]. Life-long learning considers situations in which a learner faces a series of different tasks and develops methods of retaining and using task knowledge to improve the effectiveness (more accurate hypotheses) and efficiency (shorter training times) of learning. Our research investigates methods of knowledge retention and transfer within the context of artificial neural networks and applies these methods to life-long learning problems, such as learning accurate medical diagnostic models from small samples of a patient population [15]. One of the fundamental problems in developing a life-long learning system is devising a method of retaining task knowledge in an efficient and effective manner such that it can be later used when learning a new task. We argue that this requires the consolidation of new task knowledge with previously learned task knowledge within a domain knowledge structure. This paper presents a theory of task knowledge consolidation within the context of multiple task learning (MTL) neural networks and tests that theory on a synthetic domain of tasks. The results indicate that it is possible to sequentially consolidate task knowledge provided there are sufficient numbers of training examples, sufficient internal representation (hidden nodes) in the MTL network and that task rehearsal with a small learning rate is used to overcome the stability-plasticity problem of neural networks and the catastrophic forgetting of previous learned task knowledge.

2 Background

The constraint on a learning system's hypothesis space, beyond the criterion of consistency with the training examples, is called inductive bias [8]. For example, Occam's Razor suggests a bias for simple over more complex hypotheses. Inductive bias is essential for the development of an hypothesis with good generalization from a practical number of examples. Ideally, a life-long learning system can select its inductive bias to tailor the preference for hypotheses according to the task being learned. One type of inductive bias is prior knowledge of the task domain [2]. The retention and use of task domain knowledge as a source of inductive bias remains an open problem in machine learning [21,3]. In [15, 18] we define *knowledge-based inductive learning* as a life-long learning method that uses knowledge of the task domain as a source of inductive bias. As with a standard inductive learner, training examples are used to develop an hypothesis of a classification task. However, unlike a standard learning system, knowledge from each hypothesis is saved in a long-term memory structure called domain knowledge. When learning a new task, aspects of domain knowledge are selected to provide a positive inductive bias to the learning system. The result is a more accurate hypothesis developed in a shorter period of time. The method relies on the transfer of knowledge from one or more prior secondary tasks, stored in domain knowledge, to the hypothesis for a new primary task. The problem of selecting an appropriate bias becomes one of selecting the most related task knowledge for transfer. Much of our prior work has focused on knowledge transfer and the measurement of task relatedness [19,17] for the purposes of learning a new task. The following provides a review of those aspects of knowledge transfer that that are relevant to the discussion of knowledge retention and consolidation.

2.1 Knowledge Transfer

In [18] we define the difference between two forms of knowledge transfer: representational and functional. Representational transfer involves the direct or indirect assignment of known task representation (weight values) to the model

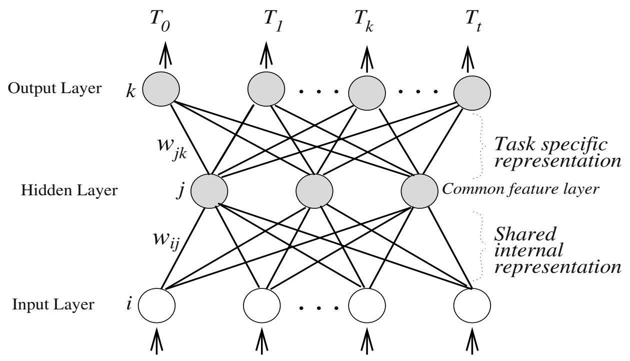


Fig. 1. A multiple task learning (MTL) network. There is an output node for each task being learned in parallel. The representation formed in the lower portion of the network is common to all tasks.

of a new task. In this way the learning system is initialized in favour of a particular region of hypothesis space within the modeling system. We consider this to be an explicit form of knowledge transfer from a source task to a target task. Since 1990 numerous authors have discussed methods of representational transfer [11,13,14,20]. Representational transfer often results in substantially reduced training time with no loss in the generalization performance of the resulting hypotheses.

In contrast to representational transfer, functional transfer does not involve the explicit assignment of prior task representation when learning a new task; rather, it employs the use of implicit pressures from training examples of related tasks [1], the parallel learning of related tasks constrained to use a common internal representation [2,3], or the use of historical training information from related tasks [21,7,9]. These pressures serve to reduce the effective hypothesis space in which the learning system performs its search. This form of transfer has its greatest value in terms of increased generalization performance from the resulting hypotheses.

Multiple task learning (MTL) neural networks are one of the better documented methods of functional transfer [2]. An MTL network is a feed-forward multi-layer network with an output for each task that is to be learned. The standard back-propagation of error learning algorithm is used to train all tasks in parallel. Consequently, MTL training examples are composed of a set of input attributes and a target output for each task. Figure 1 shows a simple MTL network containing a hidden layer of nodes that are common to all tasks. The sharing of internal representation is the method by which inductive bias occurs within an MTL network. MTL is a powerful method of knowledge transfer because it allows two or more tasks to share all or part of internal representation to the extent to which it is mutually beneficial. The more that tasks are related the more they will share representation and create positive inductive bias.

2.2 Selective Transfer and Task Rehearsal

Lacking a method of knowledge transfer [3,21] that distinguishes knowledge from related and unrelated tasks, we developed one [18,15]. ηMTL, a modified version of MTL, was created to provide a solution to the problem of selective transfer of secondary task knowledge. Using a measure of secondary task to primary task relatedness an ηMTL network can favourably bias the induction of a hypothesis for a primary task.

In [19] the *task rehearsal method* was introduced as a knowledge-based inductive learning system that is able to retain and recall learned task knowledge. Building on the theory of pseudo-rehearsal [12], previously learned but unconsolidated task representations are used to generate *virtual examples* as a source of functional knowledge. After a task T_k has been successfully learned (to a specified level of generalization error), its hypothesis representation is saved in domain knowledge. This representation acts as a surrogate for the space of input-output examples that defines task T_k. Virtual examples of the input-output space for T_k can be produced (with the same level of generalization error) by passing inputs to the domain knowledge representation for T_k and recording the outputs. When learning a new task, T_0, the domain knowledge representations for tasks $T_1...T_k...T_t$ are used to generate corresponding *virtual* output values from the set of T_0 training examples. The resulting set of virtual examples is used to relearn or *rehearse* the domain knowledge tasks in parallel with the learning of T_0 in an MTL or ηMTL network. It is through the rehearsal of previously learned tasks that knowledge is transferred to the new task.

In [17] we turned our attention to the representational form of knowledge transfer and present a theory that is based on the existence of an MTL network that contains all previously learned task knowledge in a consolidated representational form. This consolidated representation is used as the starting point for learning a new task. Task rehearsal is used to ensure the stability of related secondary task knowledge within the MTL network and stochastic noise is used to create plasticity in the unrelated task knowledge portions of the network so as to allow the new task to be learned. Consequently, beneficial knowledge transfer (inductive bias) occurs from related tasks as knowledge of unrelated tasks is lost.

The representational transfer method depends on the existence of a consolidated MTL network containing all previously learned task knowledge. The question of how new task knowledge can be consolidated into an existing neural network without loss of prior knowledge is interesting and challenging. In fact, it is the stability-plasticity problem originally posed by [4] taken to the level of learning sets of tasks as opposed to learning sets of examples. The stability-plasticity problem points out the difficulty in trying to learn a new example within a neural network while at the same time trying to maintain knowledge of previously learned examples. The loss of the previously learned knowledge has been referred to as catastrophic forgetting [6]. The following section outlines the criteria for the consolidation of task knowledge.

3 The Need for Consolidated Domain Knowledge

Long-term knowledge retention is necessary for a knowledge-based inductive learning system, however, it is not sufficient. We propose that domain knowledge must be integrated in a systematic fashion for the purposes of efficient and effective retention and for more efficient and effective transfer during future learning. The process of integration we define as *consolidation of task knowledge* and the resulting representation we define to be *consolidated domain knowledge*. More specifically, we propose the following as the major criteria for the consolidation of task knowledge:

Effective Storage: Consolidation should provide effective long-term storage of task knowledge such that the accuracy of both new and prior knowledge is not lost. In fact, consolidation should promote an increase in the accuracy of prior task knowledge as new and related tasks are learned.

Efficient Storage: Consolidation should provide an efficient method of task knowledge storage both in terms of space and time. To some extent the knowledge acquired at different times will overlap and support each other. It would be inefficient to retain duplicate copies of knowledge for the same or similar tasks. A life-long learning system must have a mechanism for integrating task knowledge into a domain knowledge structure of finite size. The method should be efficient such that the time to consolidate grows polynomially in the number of tasks retained.

Effective Retrieval: Consolidation is equally important for the transfer of domain knowledge when learning a new task. Consolidated domain knowledge should support a method of knowledge retrieval such that the most related task knowledge can be selected to favorably bias the learning of a new task.

Efficient Retrieval: Indexing into domain knowledge should also be efficient. Task knowledge retrieval should be rapid so as not to delay the learning of a new task. Consolidated domain knowledge should support a retrieval method that grows polynomially in the number of tasks retained.

4 Consolidation through MTL and Task Rehearsal

Consolidation using a connectionist network was first proposed in [5]. The report suggests a method by which the neocortex of the mammalian brain consolidates new knowledge without loss of previous knowledge. Consolidation occurs through a slow process of interleaved learning of new and old knowledge within a long-term memory structure of sparse representation. We propose that MTL and the task rehearsal method provides a mechanism for sequentially consolidating task knowledge. Each of the following sections addresses one the criterion for consolidation in an effort to make our case.

4.1 Effective Storage

An MTL network would seem to be a good choice for a long-term memory structure in which to consolidate task knowledge. Provided sufficient training examples and sufficient internal representation, an MTL network can simultaneously learn a variety of tasks of a domain without loss of accuracy [3]. Furthermore, the accuracy of related tasks can be increased within an MTL network via the learning of shared internal representation at the common feature layer [2]. However, there are two problems that must be overcome if MTL networks are to be used to sequentially consolidate domain knowledge: (1) preventing the catastrophic forgetting of previously learned tasks already existing within the MTL network, particularly unrelated tasks, and (2) avoiding high-magnitude weight representations that frustrate the learning of new internal features.

The problem of catastrophic forgetting can be solved by using task rehearsal. As shown in [17], training examples for a new task can be used to modify redundant representation in a large MTL network while virtual examples are rehearsed to maintain the existing hypotheses of related tasks. For consolidation this approach must be extended to ensure that the hypotheses for unrelated tasks are also maintained. We propose the use of a large MTL network with plenty of internal representation (hidden nodes), a small learning rate and a large number of virtual training examples for all tasks. As an upper bound, the MTL network should have sufficient internal representation for learning each task independent of all others. In practice, less internal representation will be needed because the point of consolidation is to create shared internal representation. A small learning rate will encourage the creation of features within the representation that are useful to many tasks. A large number of training examples is required during task rehearsal to ensure the accuracy of the prior MTL task knowledge as well as the integration of new task knowledge. The number and selection of examples should reflect the probability distribution over the input space as observed for the domain. Large training sets for all tasks are possible because virtual examples can be generated from the existing consolidated MTL representation of prior tasks and the newly generated hypothesis of the primary task.

The second problem, identified in [10], concerns avoiding high magnitude connection weights within a previously trained network. We addressed this issue in [17] where a consolidated MTL network is used as the starting point for learning a new task from an impoverished set of training examples. The solution employed stochastic noise to provide a degree of plasticity in the network so as to escape from the local minimum created by the high-magnitude weights. Subsequent experimentation [16] using task rehearsal and consolidated MTL networks has shown that there is no need for stochastic noise provided there is a large set of training examples for both the new and prior tasks, a small learning rate, and a method of early stopping to prevent over-fitting the model to the training data. A large MTL training set and small learning rate will ensure the maintenance of accurate hypotheses for the prior tasks and the development of parsimonious internal representation for the new task with minimal growth of network weights. Over-fitting is the major cause of high magnitude weights in a neural network. The experiments presented in this paper use a simple approach that stops training based on training set error. In future work we intend to explore a weight cost term in the back-propagation algorithm and the use of validation sets for all tasks.

In summary, we propose that consolidation can be accomplished within an MTL network provided that: (1) task rehearsal is used to maintain prior task knowledge while knowledge of each new task is integrated, (2) there is sufficient training examples so as to ensure that features are maintained and created for all tasks, (3) there is sufficient internal representation within the network for learning each task independent of all others, (4) learning occurs slowly (small learning rate) so as to increase the probability of creating internal features that are useful to all tasks and (5) there is a method of early stopping to prevent the over-fitting of new task knowledge and the creation of high magnitude weights.

4.2 Efficient Storage

MTL networks provide an efficient representational form for the consolidation of knowledge from multiple tasks. If tasks of a domain have 10 input attributes and a neural network with 5 hidden nodes and 1 output node is sufficient to learn each task, 61 connection weights would be needed to represent the network. If 20 tasks from the domain are learned and retained in separate networks then a total of 1220 weights would be required. If an MTL network with 20 hidden nodes and 20 outputs is sufficient for accurately representing the 20 tasks, a total of 640 connection weights is all that is needed. The hidden nodes can increase to 38 before the number of weights of an MTL network would exceed that of all single task networks. Network representation provides significant space savings over the original training examples. For example, just 40 training examples for an MTL network with 10 inputs and 20 target values would require 1200 values.

The time required to consolidate each new task into an ever-growing MTL network would seem to continually increase. The large number of training examples required, the small learning rate and the growing number of tasks that must be rehearsed, suggests that consolidation will require more iterations of the back-propagation algorithm for each task. However, we predict that the average time to consolidate a new task into an existing MTL network will decrease as the number of previously consolidated tasks increases. This is because the probability of the MTL network already having the required internal representation (common features) for a new task increases as the number of previously consolidated tasks increases. This will keep the time complexity for consolidation tractable in the number of retained tasks.

4.3 Effective Retrieval

Our previous research into knowledge transfer [15] has shown that the most effective methods of determining the relatedness between two tasks are structural measures that calculate a task's use of a common internal representation. MTL networks provide a common internal representation and the degree to which tasks share features of this representation have been shown to be powerful measures of task relatedness [19]. Information theoretic measures such as mutual information can compute the degree to which the hidden node features within a consolidated MTL network would be beneficial to the classification of examples of a new task.

4.4 Efficient Retrieval

We are actively working on structural measures of relatedness based on statistics such as the mutual information of task outputs with respect to the features generated at the hidden node layer. To compute such a measure between a new task and any prior task in consolidated domain knowledge requires passing the training examples through the network, recording a series of values and then computing the relevant statistics. This can be achieved in time polynomial in the number of tasks stored in consolidated domain knowledge.

5 Experiments

To test our theory of sequential consolidation using an MTL network and task rehearsal we conduct a series of experiments on a synthetic domain of tasks. Our objectives are (1) to determine the optimal conditions for consolidating the knowledge of a new task within a previously consolidated MTL network containing several tasks from the same domain and (2) to show that under these conditions that it is possible to sequentially consolidate a series of tasks within an MTL network without loss of prior task knowledge.

The first experiment applies our approach by consolidating one new task within an existing consolidated MTL network of five tasks and examines the variation in prior task accuracy (based on an independent test set) as the learning rate, number of hidden nodes, maximum training set error and number of training examples are varied. From this experiment the optimal mix of the four conditions are determined for the domain under study. The second experiment studies the sequential consolidation of a series of tasks within a large MTL network. The optimal conditions from the first experiment are used in an effort to minimize the loss of prior task accuracy.

Both experiments were conducted using the Sequential Learning Research and Application System, SLRAS, developed at Acadia[1]. SLRAS is capable of single task learning, variants of multiple task learning and task rehearsal via the generation of virtual examples from retained task knowledge (consolidated or unconsolidated).

5.1 Test Domain

The Logic domain consists of six synthetic tasks where each task's output is a boolean logic function of 4 variables. Table 1 presents the entire domain of tasks.

[1] See (http://birdcage.acadiau.ca/iitrl)

Table 1. Description of the tasks of the Logic Domain. Each of the tasks is a logical expression of four of the input variables.

Task Name	Logical Expression for Task
T_0	$(a > 0.5 \vee b > 0.5) \wedge (c > 0.5 \vee d > 0.5)$
T_1	$(c > 0.5 \vee d > 0.5) \wedge (e > 0.5 \vee f > 0.5)$
T_2	$(c > 0.5 \vee d > 0.5) \wedge (g > 0.5 \vee h > 0.5)$
T_3	$(e > 0.5 \vee f > 0.5) \wedge (g > 0.5 \vee h > 0.5)$
T_4	$(e > 0.5 \vee f > 0.5) \wedge (i > 0.5 \vee j > 0.5)$
T_5	$(g > 0.5 \vee h > 0.5) \wedge (i > 0.5 \vee j > 0.5)$

The goal in designing the domain was to create a set of non-linearly separable classification tasks that shared two linearly separable features in various ways. The features used are boolean expressions of the form $(x > 0.5 \vee y > 0.5)$, where x and y are input variables. The domain has ten inputs, labelled a through j which generate 5 features. The tasks of the domain vary in their degree of relatedness to one another based on their shared use of these features. For example, T_5 shares the feature $(g > 0.5 \vee h > 0.5)$ with tasks T_2 and T_3 and feature $(i > 0.5 \vee j > 0.5)$ with T_4. Because all tasks are non-linearly separable, at least two hidden nodes are required to form an internal representation of the features for each task. For each task, 2 hidden nodes form a disjunction of two inputs and then an output node forms a conjunction of 2 hidden nodes. This suggests that the minimum configuration of an MTL network for consolidating all tasks would be 10 input, 5 hidden and 6 output nodes.

T_0 is the new task that will be consolidated into an existing MTL in the first experiment. T_0 shares the feature $(c > 0.5 \vee d > 0.5)$ with tasks T_1 and T_2 and is thereby related to these tasks. No other task uses the feature $(a > 0.5 \vee b > 0.5)$, which necessitates that consolidation must develop this feature within the CDK MTL network in order to learn T_0.

5.2 General Method

The MTL neural networks used in the experiments have an input layer of 10 nodes, one hidden layer (common feature layer) of nodes, and an output layer of 6 nodes, one for each task. The number of hidden nodes will vary depending on the experiment from 1 to 80. In all experiments, the mean square error (MSE) cost function is minimized by the back-propagation algorithm that uses a momentum term. The learning rate varies between 0.5 and 0.001 depending on the experiment with the momentum term consistently set at 0.9. Prior to consolidation the weights of all MTL networks are randomly initialized to values in the range -0.1 to 0.1. For all experiments, training proceeds for up to 100,000 iterations or until the average MSE over all tasks decreases to a specified maximum level. The network representation is saved at this point. Each experiment reports the results of 15 repetitions using different training examples and random initial weights within the network.

Performance of the methods is compared in terms of the effectiveness of maintaining the accuracy of the consolidated domain knowledge tasks within the MTL network. Effectiveness is measured as the mean percentage of correct classifications (accuracy), over all repetitions, made by the hypotheses against a 1000 example test set. In Experiment 2, efficiency of consolidation is measured as the mean number of iterations to reach the MSE error tolerance.

5.3 Experiment 1: Consolidating a New Task into an Existing Consolidated MTL Network

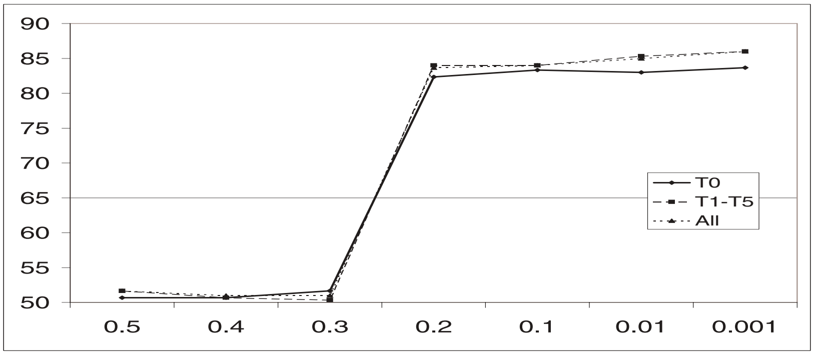
Method. This experiment examines the consolidation of task T_0 within an MTL network that is initialized with a consolidated domain representation of secondary tasks T_1 through T_5. The objective is to determine the optimal mix of conditions for best consolidation. Specifically, we are interested in maintaining the level of prior task accuracy while at the same time developing highly accurate models for T_0 (based on an independent test set). The following conditions will be varied: learning rate, number of hidden nodes, number of training examples and the maximum MSE tolerance. The maximum MSE tolerance expresses the level of average task accuracy required across all tasks being consolidated. This is used to stop training and control the growth of high magnitude weights within the consolidated MTL network.

The consolidated domain representation was produced by training an MTL network on the five secondary tasks from random initial weights using 600 training examples. Training was stopped when the MSE over all tasks dropped below a value of 0.01. An independent test set was used to estimate the generalization accuracy of these models. The average accuracy over all tasks was 86.6%.

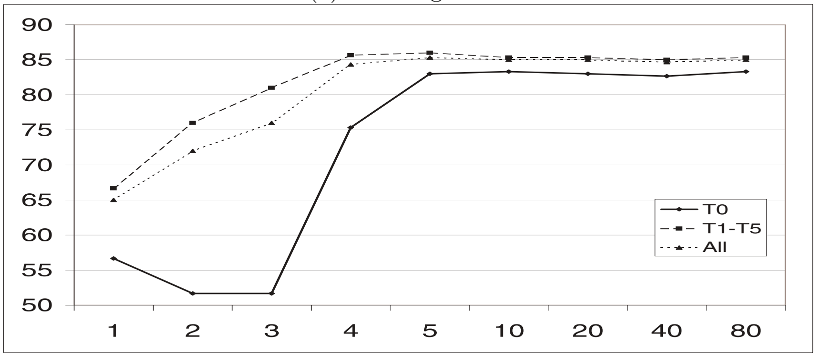
The MTL network used to consolidate T_0 has one output node added for the new task and the weights between the hidden nodes and this output node are initialized to small random values. To maintain prior domain knowledge, all secondary tasks are rehearsed within the network as the new task is learned. The existing consolidated representation of the secondary tasks is used to generate virtual examples for rehearsal by passing the training examples for the new task through the representation and producing the corresponding target outputs.

Based on preliminary experimentation, it was decided that our base values for conducting the experiment would be a learning rate of 0.01, 20 hidden nodes, maximum MSE tolerance of 0.01 and 200 randomly chosen training examples. Four studies were conducted to determine the optimal setting of each of these factors. For each study, three of the factors were fixed and the fourth factor was varied over a significant range of values.

When consolidation of T_0 begins, the errors on the previously learned tasks are very small. Only the new task shows significant error. This guides the back-propagation algorithm to find and/or create the necessary internal representation for the new task. This process will interfere with the rehearsal of the previously consolidated tasks and drive their error rates upward temporarily. However, over



(a) Learning rate.



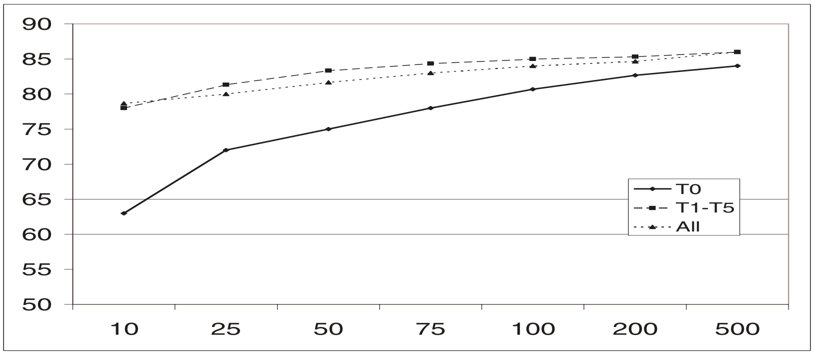
(b) Number of hidden nodes.

Fig. 2. Results of consolidating new task T_0 into an existing consolidated MTL representation. Shown is the mean percentage of correct classifications (accuracy) by the T_0 hypothesis, by T_1 through T_5 hypotheses and by all hypotheses.

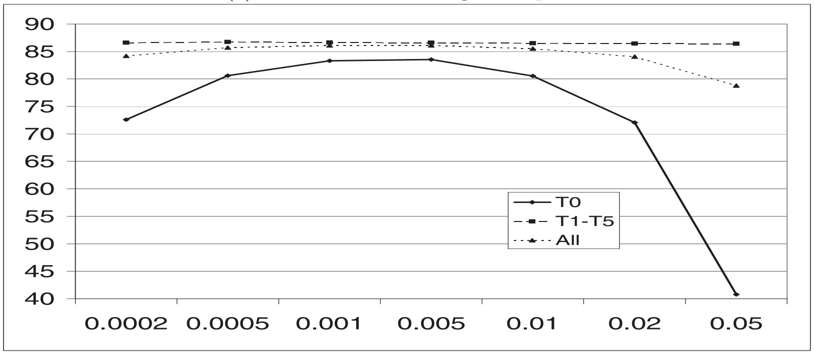
several thousand iterations, sufficient internal representation should be found for all tasks and the mean error rate should drop below the tolerance level. In this way task rehearsal is used to maintain the accuracy of prior task knowledge while the new task is consolidated into the MTL network.

Results and Discussion. Figures 2 and 3 show the results from the four studies of this experiment. Each graph shows the variation in test set accuracy as a function of (a) learning rate, (b) number of hidden nodes, (c) number of training examples and (d) maximum MSE tolerance. The results indicate that it is easy to find a range of values for each factor that allowed T_0 to be consolidated within the MTL network without significant loss of prior task knowledge.

From Figure 2 it is apparent that a learning rate less than 0.2 is sufficiently small to allow the six tasks to be consolidated. We choose a learning rate of 0.01 because it ensures that there is no statistically significant loss in prior task knowledge. Graph (b) shows that 5 hidden nodes is the minimum requirement for this domain of tasks and that increasing the number of hidden nodes beyond



(c) Number of training examples .



(d) Maximum MSE tolerance level.

Fig. 3. Results of consolidating new task T_0 into an existing consolidated MTL representation. Shown is the mean percentage of correct classifications by the T_0 hypothesis, by T_1 through T_5 hypotheses and by all hypotheses.

5 does not hinder consolidation. We choose to stay with 20 hidden nodes as this configuration provides more than sufficient representation and demands only moderate training times. As would be expected, graph (c) of Figure 3 demonstrates that the mean accuracy of the resulting consolidated models increases with the number of training examples. We choose to use 500 training examples in the subsequent experiment because of the increased accuracy it provides. The variation in mean accuracy of the models as a function of the maximum MSE on the training set is perhaps the most interesting result. Graph (d) shows that it is necessary to perform some form of early stopping so as to prevent the MTL network from over-fitting to the virtual training examples for the primary task, T_0. An optimal choice for maximum MSE tolerance is in the 0.0005 through 0.01 range. We choose the mid point of 0.005 for the subsequent experiment.

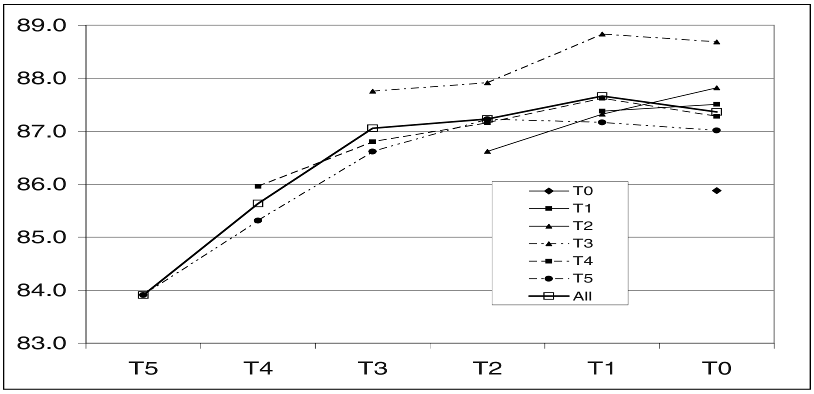
5.4 Experiment 2: Sequentially Consolidating a Series of Tasks within an MTL Network

Method. This experiment tests the consolidation method on a sequence of tasks that vary in their relatedness to one another. The objective is to show that under the proper conditions that it is possible to sequentially consolidate new task knowledge without the loss of prior knowledge and in an efficient manner. We will consider the proper conditions to be the values chosen for the four factors examined in the first experiment: a learning rate of 0.01, 20 hidden nodes, 500 training examples and a maximum MSE tolerance of 0.005.

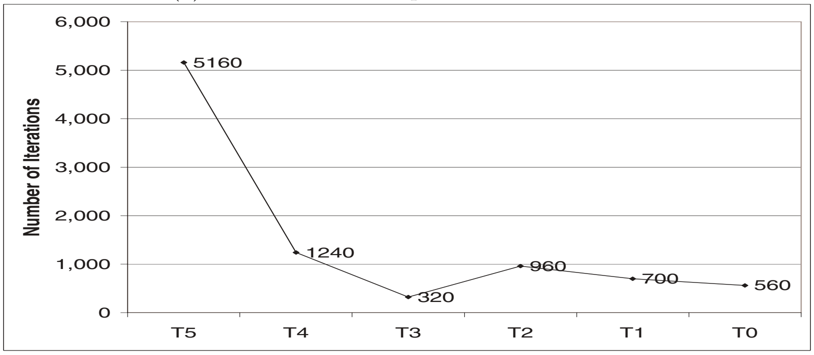
The six tasks are learned in reverse order, from T_5 through T_0. After each new task is learned the consolidated MTL network representation is saved. Before training begins on the next task, this consolidated MTL network is used as the initial representation (T_4 will be learned starting from the MTL network representation for T_5; T_3 will be learned starting with the consolidated MTL network representation for T_5 and T_4; etc). Only the weights between the hidden nodes and the new task output node are initialized to small random values. All previously learned tasks of the sequence are rehearsed within the network when consolidating a new task. The existing consolidated MTL network is used as the source of the virtual examples for this rehearsal.

Results and Discussion. Figure 4 shows the results in the order in which the tasks were consolidated. Each data series of graph (a) represents the mean percentage of correct classifications on the test set by a consolidated hypothesis at various points of the learning sequence. The graph shows that the accuracy of all tasks remains at or above their starting levels throughout the learning sequence. In fact, the performances of the consolidated representations for T_5, T_4, T_3 and T_2 increase significantly by the time the final task, T_0, is consolidated. Graph (b) supports our theory that consolidation of task knowledge within an MTL network requires fewer iterations as the number of consolidated tasks increases. Over-time the network acquires the internal representation of common features used by tasks of the domain.

The general increase in domain knowledge accuracy can be explained by two factors, where both are related to the consolidation of new task knowledge into the MTL network. First, the addition of tasks to the network act like a regularizer that prevents the network from over-fitting to the training data of any specific task. Second, there is a beneficial transfer of knowledge between related tasks as each new task is consolidated. The sharing of features created within the hidden nodes of the network provide the mechanism of knowledge transfer. For example, tasks T_5, T_4 and T_3 share three boolean logic features (see Table 1) and by the time T_3 has been consolidated the internal representation in the network has become very good at detecting each of these features. The result is that the accuracies of domain knowledge representations of tasks T_5 and T_4 increase.



(a) Effectiveness of sequential consolidation.



(b) Efficiency of consolidation.

Fig. 4. Results of the sequential consolidation of Logic domain tasks. Each line of graph (a) represents the mean percentage of correct classifications (accuracy) by a consolidated hypothesis at that point in the learning sequence. The mean accuracy over all tasks is also shown. Graph (b) shows the number of iterations required to consolidate each of the tasks of the sequence.

6 Summary and Conclusion

This paper extends the work reported in [16] that addresses a fundamental question of a life-long machine learning system: How can task knowledge be consolidated within a long-term domain knowledge structure for the benefit of future learning? We propose a theory of task knowledge consolidation that uses a large MTL network as the domain knowledge structure and task rehearsal as the method of overcoming the stability-plasticity problem and the loss of prior knowledge. Having an effective method of consolidation would be an important advance because a consolidated source of domain knowledge in representational form has been shown to provide a basis for more efficient and effective transfer of knowledge when learning a new task from sets of impoverished data [17]. Consolidated representation provides the basis for indexing into domain knowledge using deep structural measures of task relatedness and it can speed up learning through the direct use of prior representation. Experiments have been conducted on a synthetic domain of six tasks using software that was developed in accord with the theory. Under the proper conditions, the results indicate that the method is capable of sequentially consolidating task knowledge within an MTL network without loss of prior task knowledge and in an efficient manner. In fact, the experiments clearly indicate an increase in hypothesis accuracy as additional related tasks are consolidated into the MTL network. We propose that this is due to a regularization effect as well as the transfer of knowledge between related tasks. We plan to investigate this further in future work by extending the number and complexity of the tasks being consolidated and by studying real-world domains of tasks.

The proper conditions for consolidation have been found to be (1) a sufficiently large amount of training examples, (2) an abundance of internal representation (number of hidden nodes), (3) a small learning rate so as to ensure slow integration of the new task into existing representation and (4) a method of preventing the network from over-fitting and therefore creating high-magnitude weight representations. The first condition is easily met by our life-long learning system because virtual training examples can be manufactured through task rehearsal. The second and third conditions are easily met with the only impact being an increase in computational space and time for training the neural network. The last condition is not as easily dealt with. We have used a simple approach based on stopping consolidation when the training set error decreases to a predetermined level of error tolerance. This can be improved upon by employing an early-stopping technique based on a multi-task validation set. We are working on modifications to our system that will accommodate this technique.

Acknowledgement. This research has been funded by the Government of Canada through NSERC grants.

References

1. Yaser S. Abu-Mostafa. Hints. *Neural Computation*, 7:639–671, 1995.
2. Jonathan Baxter. Learning internal representations. *Proceedings of the Eighth International Conference on Computational Learning Theory*, 1995.
3. Richard A. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.
4. Stephen Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 11:23–64, 1987.
5. James L. McClelland, Bruce L. McNaughton, and Randall C. O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Technical Report PDP.CNS.94.1*, 1994.
6. Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: the sequential learning problem. *The Psychology of Learning and Motivation*, 24:109–165, 1989.

7. Tom Mitchell and Sebastian Thrun. Explanation based neural network learning for robot control. *Advances in Neural Information Processing Systems 5*, 5:287–294, 1993. ed. C. L. Giles and S. J. Hanson and J.D. Cowan.
8. Tom M. Mitchell. *Machine Learning*. McGraw Hill, New York, NY, 1997.
9. D.K. Naik and Richard J. Mammone. Learning by learning in neural networks. *Artificial Neural Networks for Speech and Vision*, 1993.
10. Lorien Y. Pratt. Discriminability-based transfer between neural networks. *Advances in Neural Information Processing Systems 5*, 5:204–211, 1993. ed. C. L. Giles and S. J. Hanson and J.D. Cowan.
11. Mark Ring. Learning sequential tasks by incrementally adding higher orders. *Advances in Neural Information Processing Systems 5*, 5:155–122, 1993. ed. C. L. Giles and S. J. Hanson and J.D. Cowan.
12. Anthony V. Robins. Catastrophic forgetting, rehearsal, and pseudorehearsal. *Connection Science*, 7:123–146, 1995.
13. Noel E. Sharkey and Amanda J.C. Sharkey. Adaptive generalization and the transfer of knowledge. *Working paper - Center for Connection Science*, 1992.
14. Jude W. Shavlik and Thomas G. Dietterich. *Readings in Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA, 1990.
15. Daniel L. Silver. *Selective Transfer of Neural Network Task Knowledge*. PhD Thesis, Dept. of Computer Science, University of Western Ontario, London, Canada, 2000.
16. Daniel L. Silver and Peter McCracken. The consolidation of neural network task knowledge. In M. Arif Wani, editor, *Proceedings of the Internation Conference on Machine Learning and Applications (ICMLA'03)*, pages 185–192, Los Angeles, CA, 2003.
17. Daniel L. Silver and Peter McCracken. Selective transfer of task knowledge using stochastic noise. In Yang Xiang and Brahim Chaib-draa, editors, *Advances in Artificial Intelligence, Proceedings of the 16th Conference of the Canadian Society for Computational Studies of Intelligence (AI'2003)*, pages 190–205. Springer-Verlag, 2003.
18. Daniel L. Silver and Robert E. Mercer. The parallel transfer of task knowledge using dynamic learning rates based on a measure of relatedness. *Connection Science Special Issue: Transfer in Inductive Systems*, 8(2):277–294, 1996.
19. Daniel L. Silver and Robert E. Mercer. The task rehearsal method of life-long learning: Overcoming impoverished data. *Advances in Artificial Intelligence, 15th Conference of the Canadian Society for Computational Studies of Intelligence (AI'2002)*, pages 2338:90–101, 2002.
20. Satinder P. Singh. Transfer of learning by composing solutions for elemental sequential tasks. *Machine Learning*, 1992.
21. Sebastian Thrun. Lifelong learning algorithms. *Learning to Learn*, pages 181–209, 1997.

Resolvent Clause Weighting Local Search

Wayne Pullan and Liang Zhao

School of Information Technology,
Griffith University, Gold Coast,
Qld., 4215, Australia
`w.pullan@griffith.edu.au`
Phone: (07)55529002, Fax: (07)55528002

Abstract. Considerable progress in local search has recently been made in using clause weighting algorithms to solve satisfiability benchmark problems. While these algorithms have outperformed earlier stochastic techniques on many larger problems, this improvement has generally required extra, problem specific, parameters requiring fine tuning to problem domains for optimal run-time performance. In a previous paper, the use of parameters, specifically in relation to the DLM clause weighting algorithm, was examined to identify underlying features in clause weighting that could be used to eliminate or predict workable parameter settings. A simplified clause weighting algorithm, Maxage, based on DLM, was proposed that reduced the DLM's run-time parameters to a single parameter. This paper presents an algorithm, RLS, based on DLM and Maxage, which combines the use of resolvent clauses with clause weighting. In RLS, clause resolvents that contain fewer literals than their parent clauses become active in the SAT problem whenever a parent clause is false and a clause weight increase is performed. This technique reduces the overhead of resolvent clauses and also adds extra information for the local search when it is required. In this paper RLS is compared with the state of the art DLM and SAPS clause weighting local search algorithms.

Keywords: Constraints, Search

1 Introduction

The propositional satisfiability (SAT) problem is fundamental in solving many practical problems in mathematical logic, inference, machine learning, constraint satisfaction, and VLSI engineering. Theoretically, the SAT problem is the *core* of a large family of computationally intractable NP-complete problems. Several such NP-complete problems have been identified as central to a variety of areas in computing theory and engineering. Therefore, methods to solve the satisfiability problem play an important role in the development of computing theory and systems.

In this paper, as with most other work on SAT algorithms, we only consider propositional formulae in conjunctive normal form. That is, formulae of the form $F = \bigwedge_i \bigvee_j l_{ij}$ where each l_{ij} is a propositional variable or its negation. The l_{ij} are termed *literals* while the disjunctions $\bigvee_j l_{ij}$ are the *clauses* of F. The goal of all SAT algorithms is to find an assignment of the truth values to the propositional variables in F that results in no unsatisfied (*false*) clauses.

Algorithms for solving SAT problems can be divided into two categories: *complete* and *incomplete*. Complete SAT algorithms perform a systematic traversal of the search space and will always find a solution if one exists. Incomplete SAT algorithms are stochastic algorithms in that they may find a solution but, if they fail, it cannot be concluded that no solution exists.

Some of the best known incomplete SAT algorithms are local search algorithms that, while they differ in detail, all basically implement a local search strategy which starts with an initial random assignment of truth values to all propositional variables. In each subsequent search step, the algorithm selects a variable using some heuristic and negates the truth value of that variable (i.e. *true* to *false* or *false* to *true*). Variable negations (flips) are typically performed with the goal of minimising an objective function based on the currently unsatisfied clauses. These algorithms are the focus of this paper and are discussed in more detail in Section 2. The remainder of this paper is structured as follows: Section 3 describes the Resolvent clause weighting Local Search algorithm (RLS) used in this paper while Section 4 discusses the experimental method used in this study. The experimental results obtained for RLS are presented in Section 5 with Section 6 containing a conclusion and suggestions for future research.

2 Local Search

At a high level, a basic local search algorithm can be viewed as a mechanism for traversing a highly multi-dimensional hyper-surface with the objective of locating a global minima. While features of the hyper-surface may be extremely complicated, the perception of the hyper-surface by the local search algorithm is very limited in that it only knows the hyper-surface immediately adjacent to its current position and has no memory or knowledge of the hyper-surface in any other location. In fact, its knowledge is limited to, for a single step in any given direction, the rate at which the hyper-surface is changing. To efficiently traverse the hyper-surface the local search algorithm must avoid:

- **Search Cycles** which are basically some configuration of closed paths on the hyper-surface. They arise because of the presence of local minima or some combination of other hyper-surface features.
- **Unguided Travel** which occurs when the hyper-surface is locally flat (plateau) and there is no basic, under-lying guidance for the search.

Search cycles that have short path lengths can be successfully handled by mechanisms such as Tabu lists [3], which prevent re-selection of a variable before some number of other variables have been modified. However, search cycles with longer path lengths or a variety of paths are much more difficult to detect and escape from. Tabu lists can also have a beneficial effect when traversing a hyper-surface plateau as they tend to provide an underlying direction for the search.

The hyper-surface traversed by local search algorithms for SAT problems is generally that formed by evaluating the number of false clauses for each assignment of variables identified during the search. That is, the local search is performing the global optimisation problem (for a SAT problem with n variables $(x_1, \ldots, x_n)$ and m clauses $(c_1, \ldots, c_m)$):

$$min \;\; f(x_1, \ldots, x_n) = \sum_{i=1}^{m} b_i \qquad (1)$$

where $b_i = 0$ if clause c_i is *true* and $b_i = 1$ if clause c_i is *false*. At each step in the search, these algorithms evaluate the effect of negating each variable in terms of the reduction in the number of false clauses and will generally select the variable which causes the largest decrease in f. The fundamental difference in these algorithms is typically in the tie-breaking rules if more than one variable gives the best decrease, and how they handle search cycles and plateaus on the hyper-surface (random moves, random restarts and Tabu lists).

The clause weighting variant of local search traverses the weighted false clause hyper-surface that is formed by the weighted cost of a problem solution. That is, a clause weighting local search is addressing the global optimisation problem:

$$min \;\; g(x_1, \ldots, x_n) = \sum_{i=1}^{m} w_i b_i, \quad w_i \geq 1 \qquad (2)$$

where w_i is the weight associated with clause c_i. At each step in the search, these algorithms evaluate the effect of negating each variable in terms of the reduction in the weighted cost of the false clauses and will generally select that variable which causes the largest decrease in g. Clause weights act to deform the weighted false clause hyper-surface (S_g) from the false clause hyper-surface (S_f) and are typically incremented whenever a local minimum or extended plateau is found by the algorithm. This action tends to remove local minima [6] and plateaus from S_g. To prevent S_g from becoming too deformed and "rugged" (and thus losing any natural underlying guidance from S_f), a clause weight reduction mechanism is normally incorporated into the search.

Clause weighting local search algorithms tend to focus on satisfying the more difficult clauses of a SAT problem as the weights for clauses that are difficult to satisfy will, on average, be higher than those for clauses that are easier to satisfy. This will make satisfying these clauses more attractive to the algorithm and is analogous to the common problem solving heuristic of attacking the most difficult parts of a problem first, and then addressing the less constrained resources until a solution is found. It is also important to note that all global minima, corresponding to $f = g = 0$, will be in the same positions on S_g as they are on S_f. If this were not the case, clause weighting local search algorithms would be at a considerable disadvantage to non-clause weighting local search algorithms in that they would be trying to locate global minima that are moving on S_g as clause weights change (as is the case when the global minima are such that $f > 0$).

There are some inherent disadvantages in clause weighting algorithms, namely that additional, problem dependent parameters are required to control the clause weighting. These include the amount by which to increase or decrease the clause weights and at what point reweighting should occur. Also, the possibility of clause weighting cycles exists where clause weights are repetitively increased and decreased causing a search cycle in the sequence of variables to be negated.

Clause weighting local search algorithms for SAT were simultaneously proposed in [6] and [10]. Various enhancements to clause weighting followed in the mid-90s, particularly Jeremy Frank's work on multiplicative weighting and weight decay [2]. Early clause weighting algorithms avoided plateau search by adding weight to all unsatisfied clauses as soon as a plateau was encountered [6]. However, it was not until the development of the Discrete Lagrangian Multiplier (DLM) algorithm [11] that these insights were translated into significant performance improvements. The main differences between DLM and earlier clause weighting techniques are in the use of a tabu list [3] to guide the search over plateau areas, and a weight reduction heuristic that periodically reduces clause weights. The Smoothed Descent and Flood (SDF) algorithm [8] uses multiplicative weighting and a continuous renormalisation of relative weights after each increase. While SDF produced some improvement over DLM in terms of the number of variable negations required on smaller sized problems, there is a significant run-time overhead in maintaining SDF's real valued weights. SDF subsequently evolved into the Exponentiated Sub-Gradient (ESG) method [9] which has recently been improved on by the Scaling and Probabilistic Smoothing (SAPS) [5] method.

Another recent algorithm is Maxage [12], which is based on DLM and has comparable performance. Of particular significance is that Maxage has only a single problem dependent parameter as compared to DLM's fourteen. The algorithm presented in this paper, RLS, is derived from both DLM and Maxage and is described in detail in the following section.

3 RLS Algorithm

RLS (Fig. 1) is an additive clause weighting local search algorithm with a single run-time parameter, DELAY, which specifies how many clause weight increase cycles must occur before a clause weight reduction is performed. As with the run-time parameters for other clause weighting algorithms, DELAY is problem dependent and must be pre-determined for each particular problem class. The three major heuristic components of RLS are:

Resolvent Clauses. The use of resolvent clauses for local search was first proposed in the Adding New Clauses (ANC) algorithm [1] where, at a local minima, new resolvent clauses of both the unsatisfied clauses and randomly selected neighbouring clauses were added to the problem. The method used in RLS is somewhat different in that, at the start of the RLS algorithm, a Resolvent Clause Cache (RCC) is created by:

- Generating all possible resolvent clauses from the original clauses of the SAT problem and adding, to the RCC, those resolvent clauses whose number of literals is less than that of the parent original clauses.
- Generating all possible resolvent clauses from the resolvent clauses in the RCC and adding, to the RCC, all those clauses whose number of literals is less than or equal to that of the parent RCC clauses.
- Associating, with each original clause of the SAT problem, a list of resolvent clauses that were generated from the original clause or from a resolvent clause that was generated from the original clause.

```
procedure RLS
begin
    Generate a random starting point
    Initialise counters and clause weights to zero
    Generate cache of resolvent clauses
    while solution not found and $flips <$ maxFlips do
        $B \leftarrow$ set of best weighted cost single flip moves
        if no improving $x \in B$ then
            if zero cost $x \in B$ and $rand() \% 100 \leq P_{flat}$ then
                $B \leftarrow$oldest zero cost $x$
            end if
        end if
        if $B \neq \emptyset$ then
            Randomly pick and flip $x \in B$
        else
            Increase weight on all false clauses and child resolvent clauses
            if ++$increases$ % DELAY = 0 then
                Decrease weight on all weighted clauses
            end if
        end if
    end while
end
```

Fig. 1. The RLS Algorithm

The resolvent clauses in the RCC only become active in the local search whenever the parent clause is *false* and an increase in clause weights is performed. The clause weights for active RCC clauses are increased / decreased in the same manner as the original clauses in the SAT problem with the exception that the lower limit for RCC clause weights is zero as compared to one for the original clauses. This technique reduces the overhead of resolvent clauses and also adds extra information for the local search when it is required. Restricting the resolvent clauses to only those containing fewer literals than their parent clauses maximises the information contained in the resolvent clause and also prevents a possible "explosion" of resolvent clauses.

Flat Move. The Flat Move heuristic ensures that, if no improving variables are available, the least recently flipped variable in the false clauses which causes no overall change in g (i.e. zero cost) has a probability $P_{flat}/100$ of being selected. Other experiments, not reported here, have shown that even with a local search that has good mobility, it is still beneficial to include some mechanism which ensures that all variables in false clauses have some probability of being selected. Those experiments also showed that RLS is robust with regard to the value of P_{flat} and a value of 10 is currently being used in RLS.

Clause Weighting. Additive clause weighting is used in RLS with clause weights increased whenever there is no improving weighted flip available and no variable is selected by the Flat Move heuristic. At each clause weight increase, one is added to the clause weight of all false clauses and their associated resolvent clauses. After

DELAY number of clause weight increase cycles have been performed, all clause weights are decreased by one, with a lower limit of one for all original clauses and zero for all resolvent clauses. The effect of the DELAY parameter can be summarised as follows: when DELAY is very close to one, clause weighting has little effect as clause weights are rapidly returned to their default value. That is, g tends to be very close to f and there is relatively little influence on the search from clause weighting. When DELAY is large, clause weights become relatively large as the downward re-weighting of clauses is performed less frequently. This tends to make the difference between f and g more significant and g becomes a more "rugged" function than f in the sense that there is greater potential for the change in g to be larger than that for f at each step of the search.

4 Experimental Method

The overall objectives of the experimental method in this paper are to:

- investigate the usability of RLS.
- evaluate the contributions of the Resolvent Clause and the Flat Move heuristics to the performance of RLS.
- compare the performance of RLS with other, state of the art, clause weighting local search algorithms.

In the following sections we provide a rationale for our choice of test SAT problems, identify test metrics, and describe our experimental design.

4.1 SAT Problem Selection

As discussed in Section 2, a local search algorithm is basically traversing a highly multi-dimensional hyper-surface to find a global minimum. To effectively test the mobility and coverage of the algorithm, test SAT problems are required such that the hyper-surfaces traversed have a range of characteristics (e.g. from "smooth" to "rugged"). In addition, the test problems should have reasonable computational requirements so that all heuristics become relevant factors in the search. Investigations, described below, have shown that the parity learning, random 3-SAT, blocks world planning and graph colouring classes of SAT problems from SATLIB[1] and DIMACS[2] benchmark libraries provide a suitable range of hyper-surface structure. As benchmark test problems for these problem classes we selected *par16-1-c* for parity learning, *f3200* for random 3-SAT, *bw_large.d* for blocks world planning and *g125.17* for the graph colouring problem. Random sampling of S_f was performed for the benchmark test problems and the results are presented in Fig. 2. This random sampling was performed by first generating uniformly random assignments for variables and then evaluating the change in f, denoted by f_i, as each variable i is negated. This process was repeated 1,000,000 times for each problem. As can be seen, the results justify the selection of these test problems as:

[1] http://www.intellektik.informatik.tu-darmstadt.de/SATLIB/
[2] ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/benchmarks/cnf/

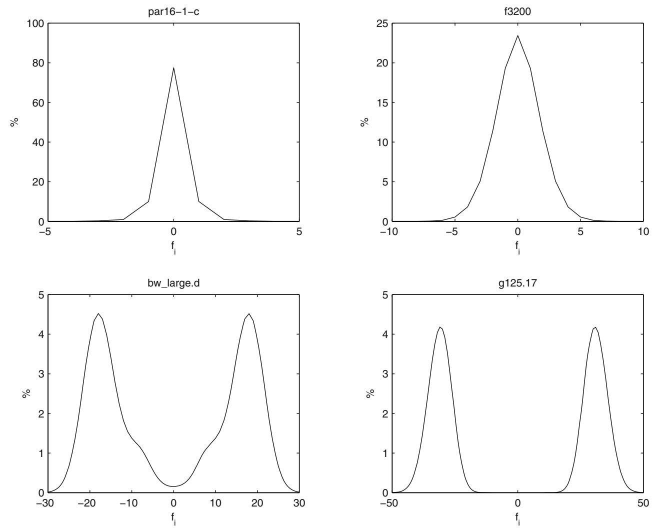


Fig. 2. Distributions of sampled f_i for each of the four benchmark test problems. This is a measure of the "ruggedness" of the f hyper-surface with *par16-1-c* being very smooth (a high proportion of zero cost flips) and *g125.17* the most rugged (no zero cost flips).

- *par16-1-c* contains a larger proportion of zero cost flips (78% of variable negations resulted in $f_i = 0$) and $|f_i| < 2$ in all other areas.
- *f3200* contains a moderately small proportion of zero cost flips (23% of variable negations resulted in $f_i = 0$) and $|f_i| < 6$ in all other areas.
- *bw_large.d* has virtually no zero cost flips and consists of moderately large features where $|f_i| \leq 30$.
- *g125.17* has no zero cost flips and consists only of large features where $15 \leq |f_i| \leq 50$.

4.2 Test Metrics

Our major focus is on the usability and performance of RLS. While other aspects of local search algorithms are important, we would rank these two aspects as the most important and they are now discussed.

Algorithm Usability. Ideally, a local search algorithm should be able to solve SAT problems without the need for the user to perform any "tuning" of the algorithm to the problem. That is, the algorithm requires no run-time parameters and is able to adapt itself to the nature of the SAT problem. However, in the absence of this ability, the number and sensitivity of run-time parameters that need to be set by the user become the most important usability issues for the algorithm. As a general rule, having fewer parameters is preferable however, there is a trade-off between the number of parameters and the sensitivity and possible ranges for these parameters. For example, if an algorithm has only a single run-time parameter but this requires tuning to all problems rather than classes of problem then this algorithm could reasonably be classed as less usable than one which has more parameters but they only require tuning to each problem class. As RLS has only a single run-time parameter, we need only investigate the sensitivity of RLS to this parameter for both particular problems and problem classes. For RLS to be classed as a usable algorithm we require that there is a wide range of DELAY values that are effective for a particular problem and that the optimal DELAY value is relatively constant for a problem class. As a side issue, these characteristics would also simplify the implementation of an adaptive version of RLS where the optimal value for DELAY is programmatically derived from an analysis of the SAT problem and automatically adjusted as the search proceeds.

With regard to the optimal value for DELAY, our hypothesis is that a larger value of DELAY (i.e. more clause weighting) will be optimal for problems where S_f is relatively smooth and a smaller value optimal when S_f is relatively rugged. Intuitively, as traversing a predominantly flat S_f takes some number of variable assignments, clause weighting needs to be more aggressive (higher DELAY) to prevent unguided travel. Conversely, if there is a small proportion or no flat areas on S_f, the main role for clause weighting is escaping from search cycles, in particular local minima, which is a relatively localised phenomena of S_f and takes relatively fewer variable assignments. Accordingly, clause weighting needs to be less aggressive (lower DELAY) so that S_g stays close to S_f and the inherent features of S_f are used as much as possible during the search.

Algorithm Performance. The performance metrics used to classify algorithms should be complete in that they measure all of a particular aspect of the algorithm. For example, as a measure of computational effort or run-time, median / average number of flips has been used. However, this measure does not effectively compare algorithms that perform different amounts of clause weighting or where the clause weighting has differing computational overheads.

Our view is that the use of processor time as a measure of computational effort provides a more encompassing metric as it reflects the total amount of work performed by the search. However, this metric does have the disadvantage that the processor time is clearly dependent on the computer processor used for the test and this makes comparison between algorithms tested on different computers difficult. To overcome this, we use the method proposed in the COCONUT [3] project and benchmark our computer processor in terms of the processor time taken to evaluate the *shekel5* function at 1.0E+8 points [4]. While this is only a basic measure of a computer processor, we feel it is adequate for comparing processor time requirements for local search algorithms.

Another potential disadvantage of using computer processor time is that it is actually a measure of the efficiency of the algorithm plus the efficiency of the implementation. The only effective solution to this problem is to encode all algorithms using identical

[3] http://www.mat.univie.ac.at/ neum/glopt/coconut

[4] A C++ program is available at *http://www.mat.univie.ac.at/ neum/glopt/coconut/shekel5.cpp*

programming techniques. As an on-going effort we are currently investigating the feasibility of doing this for the more effective local search algorithms.

In this paper we present our performance results using the Run Length Distribution (RLD) and Run Time Distribution (RTD) techniques described in [4]. While this technique has a relatively high computational cost for experiments, it does tend to eliminate the need for setting arbitrary cut-off limits which can produce mis-leading results [4].

4.3 Experimental Design

RLS Usability. With regard to usability of RLS, we need only address the sensitivity of RLS with regard to the single run-time parameter DELAY. This will be done for each benchmark test problem and also for other problems in the four benchmark problem classes. The following two experimental steps will be performed for the four benchmark test problems:

- DELAY-Range Step: using a cut-off limit of 5,000,000 flips for 100 trials with random starts, find the range(s) of DELAY which gives a success rate of greater than 80%.
- Optimal-DELAY Step: using a cut-off limit of 100,000,000 flips for 100 trials with random starts, find the value of DELAY within the range(s) identified in the DELAY-Range step which gives the dominant RTD plot [4].

The DELAY-Range step provides an overview of the sensitivity of RLS to the DELAY parameter and identifies ranges for which more intensive investigation should be performed. The Optimal-DELAY step provides a more detailed view of the sensitivity of RLS to the DELAY parameter and allows selection of the optimal DELAY value.

The DELAY-Range step will also be performed for other problems within each benchmark problem class to evaluate the sensitivity of DELAY to variations within the problem class. As the RLS heuristics need some number of weight increase / decrease clause weight cycles to be performed before the effect of DELAY can be identified, only problems that require a median number of flips greater than 100,000 will be used.

Heuristic Performance. The contributions of the Resolvent Clauses and Flat Move heuristics to the performance of RLS will be evaluated using four RTD experiments which evaluate all possible combinations of these two heuristics with the RLS Clause Weighting heuristic.

RLS Performance. The performance of the RLS algorithm, relative to two state of the art local search algorithms DLM [13] and SAPS [5], will be evaluated using the RLD / RTD method for the benchmark test problems. To improve the validity of the RLD comparisons, the SAPS program code will be modified so that it counts variable flips in the same manner as DLM and RLS (i.e. so that it does not count an increase / decrease weight cycle as a variable flip).

The value for DELAY for RLS used in this paper will be that experimentally determined in the RLS Usability experiment. The run-time parameters to be used for DLM and SAPS are shown in Table 2. The DLM parameters are as described in [13] while the SAPS parameters for each of the problem classes were found by considering SAPS's performance, over ten runs, for the 192 combinations of parameter values shown in Table 1. The choices for α (default value 1.3) consider adding almost no weight (1.01) through to settings around the default value as well as two settings (1.5 and 2.0) significantly above the default value. The small increments used at the lower end of the settings are because it was considered that even though settings of 1.01 and 1.02 represent adding very little weight, 1.02 is still adding twice as much weight as 1.01. A similar argument explains the choices for ρ (default value 0.8) except that, due to the nature of the parameter having a range from 0 to 1, small increments were used at the higher end of the scale as well.

Ten runs for each combination of α and ρ with a cutoff of 5,000,000 flips were sufficient to identify the best areas for each problem (*par16-1-c*, *f3200*, *bw_large.d* and *g125.17*). Although for some problems the success rates for even the best parameter settings did not approach 100%, areas were still identified where the majority of instances were solved within the 5,000,000 flips.

Table 1. All possible combinations of the α and ρ values shown in this table were evaluated to identify the optimal SAPS parameters for the four benchmark test problems.

Parameter	Values
α	1.01, 1.02, 1.05, 1.10, 1.15, 1.20 1.25, 1.30, 1.35, 1.40, 1.50, 2.00
ρ	0.01, 0.02, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 0.95, 0.99

Table 2. SAPS and DLM parameters used for the four benchmark test problems.

Problem	SAPS		DLM	Problem	SAPS		DLM
	α	ρ			α	ρ	
f3200	1.2	0.5	para2	par16-1-c	1.2	0.95	para4
bw_large.d	1.1	0.15	para3	g.125.17	1.15	0.10	para3

5 Experimental Results

All experiments in this paper were performed, using the methods described in Section 4.3, on a computer that required 269.33 processor seconds to execute a non-optimised version of *shekel5*, compiled under Linux using the g++ compiler.

5.1 RLS Usability

From Fig. 3 there is a wide range in the sensitivity of RLS to the DELAY parameter. For *f3200* there is a relatively sharp peak which broadens for *bw_large.d* and *g125.17* and becomes wider for *par16-1-c*. That is, RLS appears to be relatively more sensitive to the value of DELAY for problems that have a "rugged" hyper-surface and less sensitive when the hyper-surface becomes "smoother".

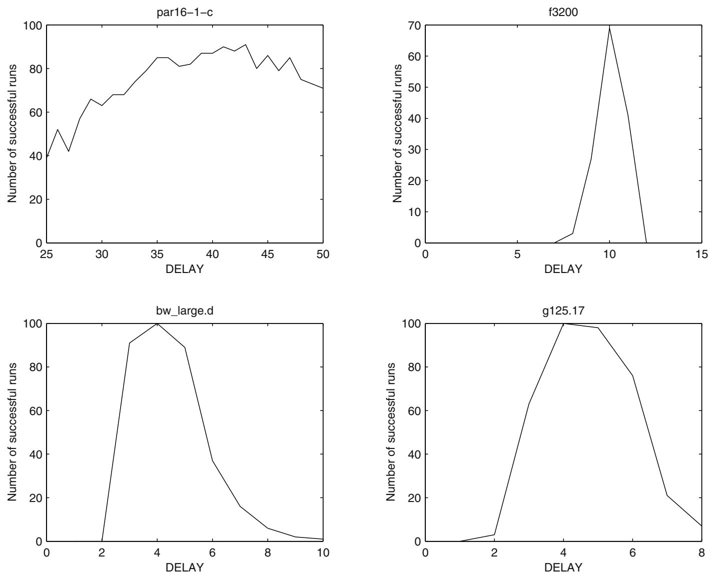


Fig. 3. Number of successful runs of RLS, from 100 attempts with random starting points, as a function of DELAY for the four benchmark test problems.

The results shown in Fig. 3 tend to confirm the hypothesis stated in Section 4.2 with regard to the optimal values for DELAY.

Fig. 4 shows the RTD plot for each of the four benchmark test problems as DELAY is varied within the critical ranges identified from Fig. 3. As would be expected, both the DELAY-Range and Optimal-DELAY steps identified identical optimal values for DELAY. Where RLS is sensitive to changes in DELAY, the curves on the RTD plot are dispersed while, when RLS is relatively robust with respect to DELAY, the curves on the RTD plot are closer.

Table 3. Optimal values of the RLS DELAY parameter for problems in the four benchmark problem classes. To effectively use the RLS heuristics, only problems that required a median number of flips greater than 100,000 have been selected.

Problem	Optimal DELAY	Problem	Optimal DELAY	Problem	Optimal DELAY	Problem	Optimal DELAY
par16-1-c	40	f1000	9	bw_large.c	5	g125.17	4
par16-2-c	39	f1600	10	bw_large.d	4		
par16-3-c	40	f2000	10				
par16-4-c	41	f3200	10				
par16-5-c	41						

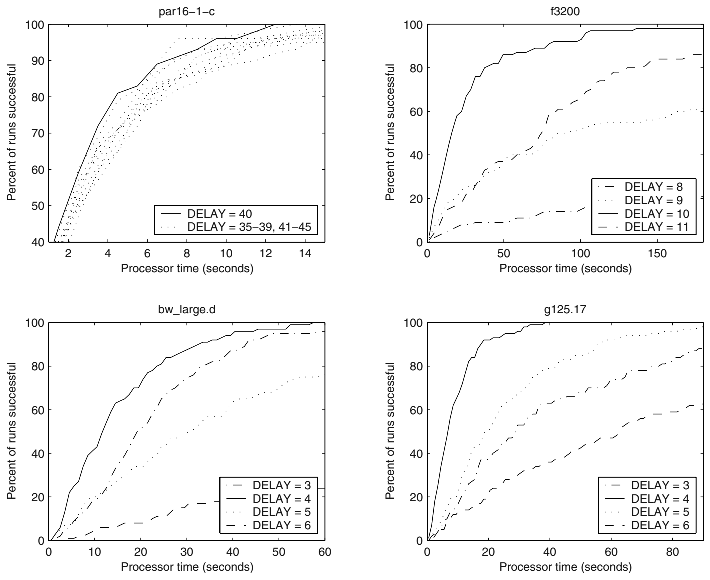


Fig. 4. RTDs for RLS, as the RLS DELAY run-time parameter varies, for each of the four benchmark test problems. The value of DELAY associated with the solid-line plot is the optimal DELAY for the problem and is the value used in this paper.

Table 3 show how the optimal value for DELAY varies within the benchmark problem classes for the more significant problems of each class. Clearly, the optimal value for DELAY is stable within each benchmark problem class.

As a general comment with regard to the usability of RLS, in addition to being stable to the DELAY within a problem class, there is also a limited range of DELAY for which RLS could be classed as bieing relatively sensitive to DELAY.

5.2 Heuristic Performance

Fig. 5 shows the effect of combinations of the RLS heuristics for the *par16-1-c* problem. The Flat Move heuristic clearly attains 100% successful runs earliest while the Resolvent Clause heuristic reduces the spread of the DELAY curves. That is, for the *par16-1-c* problem, the Flat Move heuristic tends to reduce the "heavy tail" of the distribution while the Resolvent Clause heuristic reduces the sensitivity of RLS to the DELAY parameter.

5.3 RLS Performance

Figs. 6 and 7 show, for the four benchmark test problems, the comparative performance of DLM, SAPS and RLS. With regard to the number of flips, from Fig. 6, for *f3200* and *g125.17* RLS and DLM both dominate SAPS but not each other. However, for *par16-1-c*

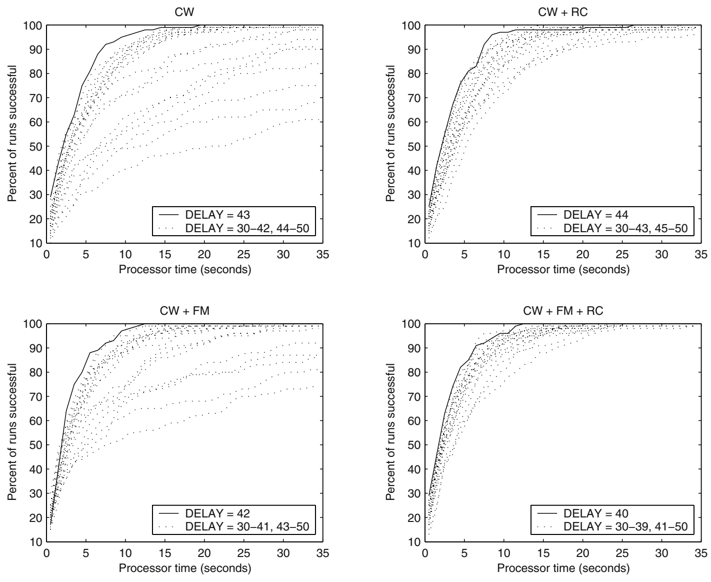


Fig. 5. RTD for RLS showing the effect of combining the Clause Weighting (CW), Flat Move (FM) and Resolvent Clause (RC) heuristics for the *par16-1-c* problem.

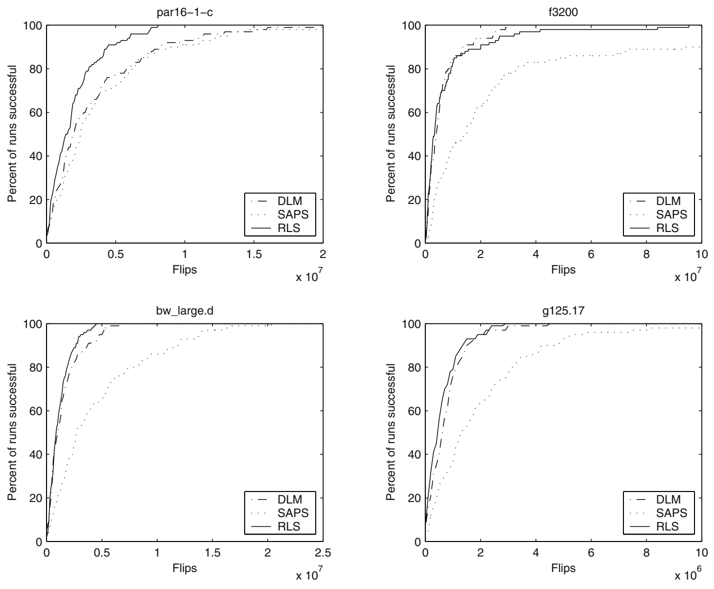


Fig. 6. RLDs for DLM, SAPS and RLS for the four benchmark test problems.

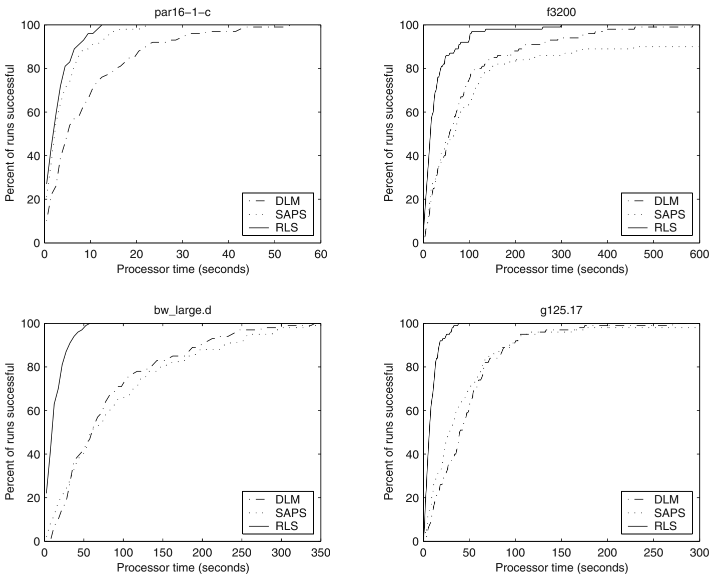


Fig. 7. RTD for DLM, SAPS and RLS for the four benchmark test problems.

and *bw_large.d*, where the number of resolvent clauses is 502 and 77 respectively, RLS is the dominant algorithm. With regard to processor time, from Fig. 7, RLS is clearly the dominant algorithm. Our intuition is that this is, with the exception of *par16-1-c* which had a large number of resolvent clauses, mainly an implementation issue between DLM and RLS. With regard to SAPS and RLS, we believe that it is partly implementation and partly algorithmic due to the relatively more complicated clause weighting techniques of SAPS.

6 Conclusion

The aim of this paper was to present and evaluate the RLS clause weighting local search algorithm. This evaluation was done with regard to both the usability and performance of RLS. Usability was investigated by evaluating the sensitivity of the RLS DELAY run-time parameter for four benchmark test problems and benchmark problem classes. The performance of RLS was evaluated by comparing it to DLM and SAPS for the four benchmark test problems. We consider RLS a step towards developing more intelligent, adaptive constraint solving technologies. Future research will include:

- the identification of structure types within SAT problems so that a run-time estimate can be programmatically made for the initial setting of the DELAY parameter [7].
- an investigation of why, for some SAT problems, there is a wide range of near optimal DELAY values, whereas for other problems it is almost unique.

- the development of adaptive control mechanisms so that the DELAY parameter can be adjusted during the search.
- an investigation to determine if clause weighting is able to identify clauses which are "globally" difficult to satisfy, or if it is a localised activity which simply gives the search adequate mobility and the global minima is arrived at without any use of "global" knowledge.
- an investigation to determine what makes clause weighting algorithms effective global optimisation algorithms. Is it in the ability to escape search cycles and the guided travel across plateaus, or is it in the initial focus on satisfying the more "difficult" clauses and then addressing the "easier" clauses?
- incorporating a parameterless clause weighting algorithm as the local search within a parallel algorithm for SAT problems.

Acknowledgment. The authors would like to thank Stuart Bain for providing the SAPS parameters used in this paper.

References

1. B. Cha and K. Iwama. Adding new clauses for faster local search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 332–337, 1996.
2. J. Frank. Learning short term weights for GSAT. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 384–389, 1997.
3. F. Glover. Tabu search: Part 1. *ORSA Journal on Computing*, 1(3):190–206, 1989.
4. H.H. Hoos and T. Stutzle. Towards a characterisation of the behaviour of stochastic local search algorithms for sat. *Artificial Intelligence Journal*, 112:213–232, 1999.
5. F. Hutter, A.D. Tompkins, and H.H. Hoos. Scaling and probabilistic smoothing: Efficient dynamic local search for sat. In *Proceedings of CP-02, LNCS 2470*, Springer Verlag, pages 233–248, 2002.
6. P. Morris. The Breakout method for escaping local minima. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pages 40–45, 1993.
7. W. Pullan, L. Zhao, and J. Thornton. Estimating problem metrics for sat clause weighting local search. In *The 16th Australian Joint Conference on Artificial Intelligence - AI03*, 2003.
8. D. Schuurmans and F. Southey. Local search characteristics of incomplete SAT procedures. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-00)*, pages 297–302, 2000.
9. D. Schuurmans, F. Southey, and R.C. Holte. The exponentiated subgradient algorithm for heuristic boolean programming. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 334–341, 2001.
10. B. Selman and H. Kautz. Domain-independent extensions to GSAT: Solving large structured satisfiability problems. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 290–295, 1993.
11. Y. Shang and B. Wah. A discrete Lagrangian-based global search method for solving satisfiability problems. *J. Global Optimization*, 12:61–99, 1998.
12. J. Thornton, W. Pullan, and J. Terry. Towards fewer parameters for sat clause weighting algorithms. In *AI 2002: Advances in Artificial Intelligence*, pages 569–578, 2002.
13. Z. Wu and B. Wah. An efficient global-search strategy in discrete Lagrangian methods for solving hard satisfiability problems. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-00)*, pages 310–315, 2000.

A Hybrid Schema for Systematic Local Search

William S. Havens and Bistra N. Dilkina

Intelligent Systems Lab
Simon Fraser University
Burnaby, British Columbia
Canada V5A 1S6
{havens, bnd}@cs.sfu.ca

Abstract. We present a new hybrid constraint solving schema which retains some systematicity of constructive search while incorporating the heuristic guidance and lack of commitment to variable assignment of local search. Our method backtracks through a space of complete but possibly inconsistent solutions while supporting the freedom to move arbitrarily under heuristic guidance. The version of the schema described here combines minconflicts local search with conflict-directed backjumping. It is parametrized by a variable ordering relation which controls the order in which the search space is explored. Preliminary experimental results are given comparing two instances of the schema to forward checking with conflict-directed backjumping [17] (*FC-CBJ*).

1 Introduction

Constraint Satisfaction Problems (CSPs) are ubiquitous in scheduling, planning, configuration and other combinatorial tasks. In general, solving CSPs is NP-hard. Modern search algorithms for CSPs are of two basic types: 1) constructive backtrack algorithms and; 2) stochastic local search algorithms. Constructive algorithms systematically expore a search tree of possible solutions [12]. They construct a consistent partial solution by extending it one variable assignment at a time and backtracking on failure, until every variable is assigned a consistent value under the problem constraints. Unfortunately, bad variable assignments made early in the search tree persist for an exponential length of time until a proof of inconsistency for a whole subtree can be found [5,11].

In contrast, local search algorithms (*e.g.* [8,14,18]) start with a complete but inconsistent assignment of variables and iteratively change individual assignments until all the constraints are satisfied. Local search is incomplete, but it is not plagued by the tyranny of early bad decisions in the search tree. These algorithms are free to move arbitrarily in the search space by following the heuristic gradient.

However, the lack of systematicity of local search algorithms makes remembering the history of past states problematic. In the worst case, a state cache exponential in the size of the CSP is required [1].

There has been recent research into hybrid search schemes which combine desirable aspects of constructive and local search methods [4,7,11,13,14,15,16]. These algorithms attempt to retain some of the systematicity of constructive search while allowing for a maximum amout of flexibility of movement in the search space. In the work reported here, we define a new hybrid scheme, called *Maximal Constraint Solving (MCS)*, which extends these methods as follows:

- The scheme operates from a nearly complete instantiation of values to variables [7,16].
- The notion of a maximally consistent solution relaxes the requirement that the constructed partial solution remain consistent [4].
- Forward checking of both assigned and unassigned variables is performed [16].
- Systematicity is enforced using a nogood cache of known inconsistent variable assignments [7,10,11,19].
- Various randomized (arbitrary) backtracking methods are supported [9,11, 13,21].
- The scheme is parameterized by a variable ordering relation which realizes multiple possible search control stuctures.

In Section-2 we give some preliminary definitions. In Section-3 we outline a novel characterization of backtrack search in order to explain the MCS schema and give an informal synthesis of the schema. Section-4 then presents a more formal characterization of the scheme which is in turn followed in section-5 by some preliminary experimental results. Finally in section-6, we briefly relate our work to other relevant research in the field.

2 Definitions

We begin with a few necessary definitions then describe the schema.

Definition 1. *A* Constraint Satisfaction Problem (CSP)*, is a triple* (V, D, C)*, where* $V = \{v_1, ..., v_n\}$ *is a set of variables with corresponding domains* $D = \{D_{v_1}, ..., D_{v_n}\}$ *and* C *is a set of k-ary constraints. A* k-ary constraint *defines the allowed tuples of values for a subset of k variables* $(v_1, ..., v_k)$ *from* V*.*

We are interested in variable assignments, written $\langle x = a \rangle$, for variable $x \in V$ and $a \in D_x$, which during search act as unary constraints that are repeatedly added and retracted. In addition, it will be useful to refer to sets of variable assignments, called *labels*, to represent environments, solutions and nogoods.

Definition 2. *A* label*,* $\lambda(X) = \{\langle x = a \rangle\}_{x \in X}$*, is a set of variable assignments, where* $X \subseteq V$ *and* $a \in D_x$*.*

A label corresponds to a snapshot of variable assignments at a particular point in the search. The current assignments of these variables may or may not correspond to the assignments specified in the label.

Definition 3. *A label, λ, is* valid *iff every variable assignment $\langle x = a \rangle \in \lambda$ is the current assignment of the variable x.*

During search we will induce nogoods, i.e. partial assignments that are not part of any consistent solution.

Definition 4. *A* nogood *is a label, $\lambda_\perp = \{\langle x = a \rangle\}_{x \in X}, X \subseteq V$, whose partial assignment of variables, X, is precluded from any global solution.*

Nogood are stored in a nogood cache, Γ, which records the infeasibility of particular variable assignments in any global solution. Now we can be more precise about which variable assignments are allowed in the current environment of other variables.

Definition 5. *An assignment, $\langle x = a \rangle$, is* disallowed *iff $\exists \lambda_\perp \in \Gamma$ s.t. $\langle x = a \rangle \in \lambda_\perp$ and $\lambda_\perp \setminus \{\langle x = a \rangle\}$ is valid. Otherwise, the assignment remains* allowed.

A variable assignment is disallowed if it is the *culprit* of a nogood whose remaining assignments constitute a valid label in the current environment. Contrary to usual practice[19,6], the culprit in a nogood can be any variable in the label. To reiterate, we are concerned only with allowed/disallowed variable assignments whether or not these assignments are consistent/inconsistent under the constraints on these variables.

Definition 6. *The* live domain *Δ_x of a variable x is its allowed elements*

$$\Delta_x = \{a \in D_x \mid \langle x = a \rangle \textit{ is allowed } \}$$

We use a heuristic valuation function to choose among the elements in the live domain of a variable.

Definition 7. *The function, $f(a)$, of a particular variable assignment, $\langle x = a \rangle$, defines its* heuristic valuation *based on the constraints on x and the current environment of other variable assignments.*

These valuations will be used to determine which domain elements constitute maximal assignments for x.

Definition 8. *$\langle x = a \rangle$ is a* maximal assignment *iff $\forall b \in D_x, f(a) \leqslant_s f(b)$.*

If every variable obtains a maximal assignment, then the label, $\{\langle x = a \rangle\}_{x \in V}$, is a *maximal solution.* No variable prefers a different assignment in the environment of the other variables. However, this maximal solution may not satisfy all the constraints in C thereby inducing a nogood on these constraints.

3 Informal Synthesis

Instead of viewing backtrack search as tree search, we consider that the current state of the search divides the variables, $\{x_1, \ldots, x_n\}$, into three (possibly empty) classes:

1. Those variables, $\{x_1, \ldots, x_{i-1}\}$, which comprise the current consistent partial solution, called the `CONSISTENT` variables. The assignment of each of these variables satisfies all the CSP constraints on this class.
2. The variable, x_i, at the fringe of the search identified as the *culprit* for backtracking is in state `INF`. Its current assignment is known to be infeasible given the environment of the `CONSISTENT` variables.[1]
3. The remaining variables, $\{x_{i+1}, \ldots, x_n\}$, are the future variables which remain unassigned and hence have state `NULL`.

Figure 1 shows these classes with the fringe of the search proceeding from left to right according to the assigned (possibly dynamic) ordering of the CSP variables. A solution has been found when every CSP variable is `CONSISTENT` and both the `INF` and `NULL` classes are empty. This simple model applies to various traditional backtracking schemes. In chronological backtracking, any inconsistency in the partial solution causes the culprit to be the `CONSISTENT` variable with the lowest precedence. In backjumping [5] and related intelligent backtracking schemes [19], when the culprit is identified within the `CONSISTENT` variables, then every other variable in this class but higher in the ordering is returned to the unassigned `NULL` category.

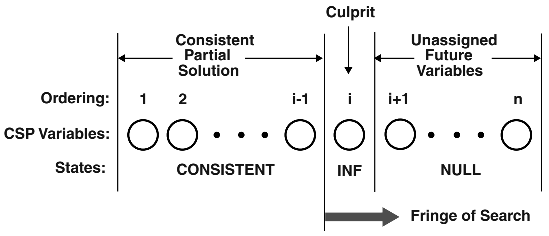
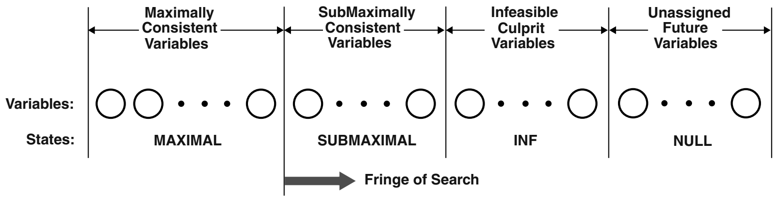


Fig. 1. The three possible states for backtrack search algorithms

But constructive algorithms will tour a significant fraction of the exponential search space if allowed, even when constraint propagation techniques [12] are employed. Instead practitioners rely upon variable and value ordering heuristics to improve the search efficiency. Heuristics are most informed when every variable

[1] The `INF` class is empty as search proceeds forward but instantiated during backtracking

in the CSP is assigned and least effective when no variables are assigned. In constructive search, the latter case tragically corresponds to the first variable in the search tree. Early mistakes in constructive search are exponentially expensive but must be made with the least available information for heuristic choice. The very mechanism of backtracking is self defeating by being hostile to the use of effective heuristics to guide the search. What we desire is a new synthesis incorporating the systematicity of constructive search with the heuristic effectiveness of local search.



Maximally Consistent Variables
SubMaximally Consistent Variables
Infeasible Culprit Variables
Unassigned Future Variables
Variables:
States:
MAXIMAL
SUBMAXIMAL
INF
NULL
Fringe of Search

Fig. 2. The four possible states for maximal constraint solving

In our MCS schema, instead of maintaining partial consistent assignment, we allow variable assignments that are inconsistent. The variable states for backtrack search identified previously in Figure 1 are insufficient. We require a distinction be made in Figure 1 between variables assigned maximal values and those remaining assigned variables which can choose (potentially) better allowed assignments from their live domains. Thus the class `CONSISTENT` is divided into two subclasses `MAXIMAL` and `SUBMAXIMAL` as shown in Figure 2. Variables can belong to one of four possible classes given the constraints on their variables and the nogoods which have been induced on their assignments.

1. A variable is `MAXIMAL` when its assignment is both allowed and maximal. The variable prefers the current assignment from its live domain.
2. If the current variable assignment is allowed but another better (more maximal) assignment is possible then the variable is `SUBMAXIMAL`.
3. The variable is in an infeasible state, `INF`, if its current assignment is disallowed, i.e. if the current assignment is known nogood in the current environment of other variable assignments. It must change its assignment if possible but induce a nogood otherwise.
4. Otherwise, the variable is in the `NULL` state. Either it can choose a new `MAXIMAL` assignment from its live domain or it must induce a new nogood on the environment of other variable assignments which have emptied its domain.

```
1   function $solve(V, C)$ {
2       $\alpha = \{\langle x = null \rangle\}_{x \in V}$
3       repeat {
4           while ($\alpha \neq$ MAXIMAL) {
5               let $x = select(V)$;
6               $assign(x)$;
7               if ($\lambda_{nullset} \in \Gamma$) return noSolution;
8           }
9           $\forall c \in C$ s.t. c is inconsistent {
10              $\lambda_{\perp} = label(c)$;
11              add $\lambda_{\perp}$ to $\Gamma$;
12          }
13      } until ($\alpha =$ MAXIMAL);
14      return $\alpha$;
15  }
```

Fig. 3. The *solve* algorithm finds a consistent solution of the variables V for the constraints C.

4 Systematic Local Search

In this section, we develop our approach more precisely.

4.1 Maximal Constraint Solving

Our MCS schema searches heuristically through a space of maximally consistent variable assignments while backtracking on assignments which are not acceptable solutions. It discards the maintenance of a totally consistent partial solution in backtrack search [4]. Instead, we require that every variable in the partial solution be assigned a value which is both *allowed* and *maximal.*

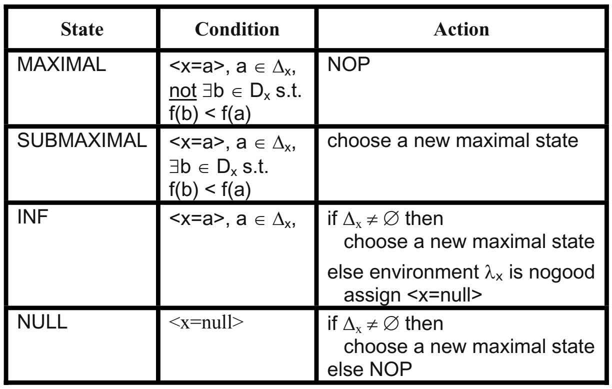
Thus, it combines desirable aspects of systematic backtrack search and heuristic local search in the same asynchronous schema [14,21]. Variables use value ordering heuristics to choose a *maximal assignment* from their live domain of allowed values. If no allowed values remain for some variable then that variable induces a nogood and backtracks. When all variables have chosen a maximal assignment, these assignments constitute a *maximal solution.* Such a solution is a mutual local maxima for every variable. If the maximal solution does not exhibit full consistency then the solution induces nogoods and again the system backtracks.

Our systematic local search algorithm is listed in Figure 3. Given a set of variables, V, and constraints, C, $solve(V, C)$ returns the first solution, α, whose assignments satisfy each constraint in C. The algorithm operates as follows. In the initial global assignment, α, (line 2) every variable is NULL. The loop beginning in line 3 is repeated until every variable assignment in α is in class MAXIMAL (in line 13). Then the solution, α, is returned in line 14. While every variable is not MAXIMAL (line 4), a variable, x, is chosen via the variable ordering heuristic, $select(V)$.[2] Then the variable, x, is assigned according to the method described in the next subsection.However, if an *empty nogood*, $\lambda_{nullset}$, is ever derived (line 7) then the algorithm returns failure indicating that no solution exists.

When every variable is MAXIMAL, the current global assignment, α, is a maximal solution but may not be a consistent solution. Beginning in line 9, for every constraint, $c \in C$, which is not satisfied, a new nogood, $\lambda_{\perp}$, is derived from c and added to the nogood cache, Γ, in line 11.[3]

4.2 Variable Assignment

We define a method, $assign(x)$, which specifies the action taken by variable, x, given the current state of that variable (according to the classification of Figure 2). The semantics of this method are given in the table of Figure 4 as state transition rules.



State	Condition	Action
MAXIMAL	<x=a>, $a \in \Delta_x$, not $\exists b \in D_x$ s.t. $f(b) < f(a)$	NOP
SUBMAXIMAL	<x=a>, $a \in \Delta_x$, $\exists b \in D_x$ s.t. $f(b) < f(a)$	choose a new maximal state

INF	<x=a>, $a \in \Delta_x$,	if $\Delta_x \neq \varnothing$ then
		choose a new maximal state
		else environment λ_x is nogood
		assign <x=null>

NULL	<x=null>	if $\Delta_x \neq \varnothing$ then
		choose a new maximal state
		else NOP

Fig. 4. Possible states of a variable, x, and consequent action given the current global assignment and live domain of variable, x, used by the method, $assign(x)$.

Variable x is in a MAXIMAL state if it is assigned $\langle x = a \rangle$ and $f(a)$ is maximal. No action is required (NOP) since the variable cannot possibly find a better (more maximal) state. Similarly, if x is in a SUBMAXIMAL state when assigned but another element, $b \in \Delta_x$, exists such that $f(b) \leqslant f(a)$. Then the action is to choose $\langle x = b \rangle$ as a new MAXIMAL state for x according to the minconflicts heuristic. However, x is in the INF state if assigned $\langle x = a \rangle$ but the value $a \notin \Delta_x$. Again the appropriate action depends on whether other elements remain in Δ_x.

[2] Variable ordering is defined immediately below.

[3] In practice, without an infinite nogood store, the algorithm stills remains incomplete since it does not systematically cover the entire search space.

If so then choose some new assignment and enter the MAXIMAL state. Otherwise, report that λ_x is nogood and undo the current assignment by assigning $\langle x = null\rangle$ thus entering the NULL state.

Finally, variable x is in the NULL state whenever unassigned. Initially, all (future) variables are NULL and variables return to this state whenever they are assigned $\langle x = null\rangle$. The appropriate action depends on whether other elements remain in Δ_x. If so then choose some new assignment and enter the MAXIMAL state. Else, do nothing (NOP) until the environment, λ_x, of other variables constraining x changes forcing x into some other state.

Nogoods force the variables away from maximal but unacceptable solutions to the CSP. It is analogous to other digression mechanisms in local search such as *tabu search*[8] where the tabu list records local maxima which cannot be revisited for some period. The difference here is that our digression mechanism is systematic. It inherits the sophisticated logical machinery of intelligent backtracking but operates within the local search paradigm.

The instance of the MCS schema presented here, *MinCon-CBJ*, uses *min-conflicts* which chooses assignments which violate the minimum number of constraints on x given the current state of the search. It is known that good value ordering heuristics can greatly improve the efficiency of both constructive and local search algorithms [20]. In general, we assume that effective heuristics are problem specific and hence a parameter of the schema. The valuation function can be redefined to compute something different than constraint violations.

Definition 9. *The* heuristic valuation*, $f(a)$, for each $a \in D_x$, is the number of constraints that disallow the variable assignment $\langle x = a\rangle$ in the current environment.*

The constraint violations are computed based on forward checking propagation. We maintain conflict counts for all variables, so that we can always ask the question how many constraints will be violated if a particular assignment were made in the current environment, similarly to Prestwich [16].

4.3 Variable Ordering

Variable ordering, *select*(V), determines the overall control structure in our approach. From Figure 2, we are reminded that variables assume four different classes corresponding to their current assignments and live domains. Thus we select variable execution order based on these classes. We define a total order among the variables by first choosing precedence among the categories SUBMAXIMAL, INF and NULL. We do not include the MAXIMAL category because executing a maximal variable does not have any effect. In addition, we choose a separate strategy within each class that imposes a total order. Both the ordering among the classes and the class strategies would influence the behaviour of the algorithm. We always choose to execute the variable lowest in the resutling total ordering.

For the *MinCon-CBJ* we experimented with two different versions of the variable ordering heuristic method, *select*(V):

1. *minCon-Heuristic* implements the precedence ordering, `NULL` $\prec$ `SUBMAXIMAL` $\prec$ `INF`, which says that we prefer a full assignment of variables and then move heuristically to make all submaximal assignments maximal before backtracking any infeasible variables. Within the `NULL` class, variables are ordered lexicographically. `SUBMAXIMAL` variables are ordered by maximum possible improvement, such as [4]. Variables in the `INF` class are not ordered but chosen randomly. Basically, this ordering is aiming at maximizing the heuristic guidance. It resorts to dealing with `INF` only after all heuristically desirable decisions have been made.
2. *minCon-Systematic* implements the ordering, `INF` $\prec$ `NULL` $\prec$ `SUBMAXIMAL`, which attempts to resolve infeasible variables before pursuing a full assignment before repairing any submaximal assignments. This method resolves nogoods first trying to always maintain an allowed partial solution. Variables within class `INF` are chosen randomly as the nogood culprit, thus achieveming diversification. Like the above method, `NULL` variables are chosen for execution lexicographically and `SUBMAXIMAL` variables by maximum improvement.

There are other interesting possibilities such as giving the `SUBMAXIMAL` class highest precedence. Also we could decide to choose among the variables in the `SUBMAXIMAL` class the variable assignment with the worst valuation rather than with the best possible improvement. Or we could choose randomly among the `NULL` variables. The numerous different instances of the variable ordering based on the four classes we have defined are very interesting. Further investigation is needed to compare their performance and evaluate their behaviour.

5 Experimental Results

The experiments described in this section are very preliminary. We generated an ensemble of solvable random binary CSPs each having 20 variables and domain size 10 around the phase transition region for probabilities p1=0.5 and p2 varying from 0.32 to 0.42 in increments of 0.01. Following the advice of [3], we used complete search (FC-CBJ) [17] to identify 200 soluble instances for each p2 value. Our results are shown in Figure 5. As expected, the hardest instances for FC-CBJ where at approximately $\kappa = 0.95$ which required the most number of backtracks (assignment changes) to find a first solution. As κ increases, we found it more and more difficult to find soluble instances.

Our *MinCon-CBJ* method performed very well on this random ensemble upto and beyond the phase transition point using either the *minCon-Systematic* or the *minCon-Heuristic* variable ordering heuristics (control strategies). Indeed, both methods outperformed *FC-CBJ* significantly on smaller values of κ where the number of solutions to the CSP is expected to be large. Our method was approximately twice as efficient as *FC-CBJ* at the transition point. Beyond $\kappa = 1.0$, *MinCon-CBJ* using the *minCon-Heuristic* control strategy still significantly dominates the complete search method. We believe the superiour performance is due to the strategy of finding a full and maximal (but perhaps inconsistent)

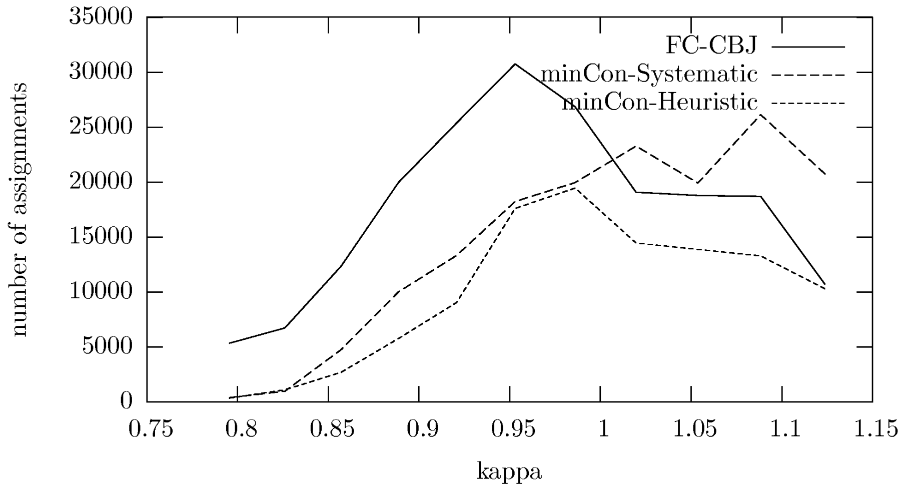


Fig. 5. Results comparing constraint-directed backjumping (FC-CBJ) against MinCon-CBJ using both minCon-Systematic and minCon-Heuristic variable ordering heuristics on random binary CSPs instances $\langle 20, 10, 0.5, p2 \rangle$.

assignment before backtracking inconsistent constraints. However, *MinCon-CBJ* using the alternate *minCon-Systematic* control strategy loses its effectiveness for high values of κ. These overconstrained problems are easy for complete search which fails whole subtrees quickly. We conjecture that they are more difficult for our second control strategy because we are backtracking incompletely through a space of full but inconsistent assignments. For high values of κ, systematicity becomes more important, allowing the extremely tight constraints to prune the search quickly.

Obviously, many other search control structures can be realized by specifying different variable ordering strategies as above. We are currently investigating some promising variants.

6 Discussion

Efforts to synthesize constructive and local search have made good progress. Recent work has focused on breaking the tyranny of systematic backtracking while maintaining a consistent partial solution. Gomes *et al* [9] randomize the identification of the culprit variable for backtracking. Prestwich [16] forgoes completeness in an incomplete dynamic backtracking (IDB) algorithm which identifies the culprit variable for backtracking heuristically (or randomly). IDB is viewed as a local search method which maintains a consistent partial assignment while attempting to maximize the number of assigned variables. Our scheme, similarly, utilizes the ability to move arbitrarily by choosing randomly or according to any other strategy among the INF variables. In addition, we retain the freedom of heuristic order of reassignment among the variables in the SUBMAXIMAL and NULL categories. Jussein and Lhomme [11] describe a hybrid framework which supports two types of moves: 1) extending a consistent partial solution which does not violate any cached nogood, otherwise; 2) repairing an inconsistent state by moving in a neighbourhood which does not violate the identified conflict nor the nogood cache. Various heuristic strategies are possible for either type of move. These two moves directly correspond to choosing among the NULL variables and choosing among the INF variables respectively, while the heuristic strategies they discuss are equivalent to the variable orderings within these two classes used in our framework.

In the instance of the MCS presented here we use *minconflicts*. Minton *et al.* [14] not only identified this natural value ordering heuristic for CSPs but developed a constructive search algorithm, called *informed backtracking*, which worked simultaneously from a consistent partial solution and total inconsistent assignment. The latter is used by minconflicts to choose value assignments during backtracking. But perhaps the most influential synthesis has been the work of Freuder and Wallace [4] which abandoned the maintenance of a consistent partial solution for soft CSPs. Their method substitutes *branch&bound* techniques for backtracking in a variety of well known constructive algorithms. They also identify the notion of a maximal solution for soft CSPs. By subsitituting the notion of a maximal assignment for consistency in partial solutions, we obtain the power of working from a full assignment similarly to Freuder and Wallace [4]. Thus value ordering heuristics have access to an artifact with as much information as possible to guide the search. While here we are presenting an algorithm for solving relational CSPs, it is easy to see possible extensions of the MCS schema to soft CSPs. As a naive modification, every time we reach a maximal solution, we could remember the one with best quality and always fail the whole solution thus allowing each individual constraints to be inconsistent in future solutions but not the exact same combination of them.

Another important and relevant research issue is the tenure policy for nogoods in the cache. In an insightful short paper, Ginsberg and McAllester [7] suggested that constructive and local search differ mainly in the memory (nogood cache) used to remember past states (failures). For systematic search algorithms the cache size can be capped. For examples, chronological backtracking requires only $O[n]$ space to remember past variable assignments but has an extremely rigid control structure. Less rigid is dynamic backtracking [6] requiring only $O[nd]$ nogood space but only slightly relaxing the systematicity of the search. Clearly, for purely local search which always follows the heuristic gradient, the number of nogoods induced can approach $O[d^n]$. However, Lynce *et al* [13], who describe a hybrid algorithm employing randomized backtracking for solving hard boolean satisfiability problems, claim this is not a problem in practice. Havens [10] shows empirical evidence that bounded nogood caches work well in distributed backtracking. Otherwise, completeness must be forsaken. In [11], a finite cache in the form of a Tabu [8] is used. Whereas [16] remembers no previous states at all which sometimes leads to deadlock. We seek to explore the middle ground of currently unknown search algorithms which maximize the use of good heuristics to guide the search while retaining only enough systematicity to limit the required size of the nogood cache. Similarly to the hybrid framework presented by Jussein and Lhomme [11], the MCS schema is parametrized and can support different nogood tenure policies. We believe that polynomial space caching schemes such as *k-relevance learning* [1] will provide the right balance between the maximum size of the cache and the ability to move freely in the search space.

7 Conclusion

We presented a new hybrid constraint solving schema which retains some systematicity of constructive search by remembering nogoods and backtracking while incorporating the heuristic guidance of local search by working on a complete and maximally consistent solution. We divide the CSP variables into four classes and define an asynchronous variable execution model that operates on these states. Various search algorithms can then be defined parametrically by specifying different variable ordering heuristics for each class allowing for arbitrary changes in variable assignment and hence freedom to move. Finally, we gave two instances of our scheme, both of which are new algorithms combining desirable aspects of both constructive and iterative search. Some interesting experimental results were presented which suggest that these methods work well. We are currently investigating additional instances and caching policies. Future work involves the extension of this framework to soft and valued CSPs using the cSemiring formalism [2] and replacing the notion of a consistent solution with that of an acceptable solution.

References

1. R.Bayardo and D.Miranker (1996) A Complexity Analysis of Space-Bounded Learning Algorithms for the Constraint Satisfaction Problem, in *proc. AAAI-96*, pp.298-304.
2. S.Bistarelli, U.Montanari and F.Rossi (1995) Constraint Solving over Semirings, in *proc. IJCAI-95*, Morgan-Kaufman.
3. D.A.Clark, J.Frank, I.P.Gent, E.MacIntyre, N.Tomov and T.Walsh (1996) Local Search and the Number of Solutions, in *proc. Principles and Practice of Constraint Programming*, pp.119-133.
4. E.C. Freuder and R.J. Wallace (1992) Partial Constraint Satisfaction, *Artificial Intelligence* 58:21-70
5. J. Gaschnig (1979) Performance measurement and analysis of certain search algorithms. Tech. report CMU-CS-79-124, Carnegie-Mellon University
6. M.L. Ginsberg (1993) Dynamic backtracking. *Journal of Aritificial Intelligence Research* 1:25-46
7. M.L. Ginsberg and D.A. McAllester (1994) GSAT and dynamic backtracking. in *proc. 4th Int. Conf. on Principles of Knowledge Representation and Reasoning*

8. F.Glover (1990) Tabu search: a tutorial, *Interfaces 20*(74-94).
9. C. Gomes, B. Selman and H. Kautz (1998) Boosting Combinatorial Search through Randomization, *proc. Fifteenth National Conference on Artificial Intelligence*, pp.431-437, Madison, WI, AAAI Press.
10. W.S.Havens (1997) NoGood Caching for MultiAgent Backtrack Search, in *proc. AAAI-97 Constraints and Agents Workshop.*
11. N. Jussein and O. Lhomme (2002) Local search with constraint propagation and conflict-based heuristics, *Artificial Intelligence* 139:21-45.
12. V. Kumar (1992) Algorithms for constraint satisfaction problems - a survey, *AI Magazine 13*(32-44)
13. I. Lynce, L. Baptista and J. P. Marques-Silva (2001) Stochastic Systematic Search Algorithms for Satisfiability, *LICS Workshop on Theory and Applications of Satisfiability Testing (LICS-SAT)*, June 2001.
14. S. Minton, M. Johnston, A. Phillips and P. Laird (1990) Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence* 58:161-205
15. P. Morris (1993) The breakout method for escaping from local minima. in *proc. AAAI-93* pp.40-45
16. S. Prestwich (2001) Local Search and Backtracking vs Non-Systematic Backtracking, *proc. AAAI 2001 Fall Symposium on Using Uncertainty within Computation.*
17. P.Prosser (199x) Hybrid Algorithms for the Constraint Satisfaction Problem, *Computational Intelligence 9(3)*:269-xxx
18. B.Selman, H.Levesque and D.Mitchell (1992) A new method for solving hard satisfiability problems, in *proc. 10th National Conf. on Artificial Intelligence,*pp.440-446.
19. R.M. Stallman and G.J. Sussman (1977) Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artificial Intelligence* 9:135-196
20. E.Tsang (1994) *Foundations of Constraint Satisfaction*, Academic Press.
21. M.Yokoo (1994) Weak commitment search for solving constraint satisfaction problems, in *proc. AAAI-94*, pp.313-318.

Constraint Satisfaction Methods for Information Personalization

Syed Sibte Raza Abidi[1] and Yong Han Chong[2]

[1] Faculty of Computer Science, Dalhousie University, Halifax B3H 1W5, Canada
sraza@cs.dal.ca
[2] School of Computer Sciences, Universiti Sains Malaysia, Penang 11800, Malaysia
thestep@cs.usm.my

Abstract. Constraints formalize the dependencies in a physical world in terms of a logical relation among several unknowns. Constraint satisfaction methods allow efficient navigation of large search spaces to find an optimal solution that satisfies given constraints. This paper explores the application of constraint satisfaction methods to personalize generic information content with respect to a user-model. We present a constraint satisfaction based information personalization framework that (a) generates personalized information via the dynamic selection and synthesis of multiple information-snippets; and (b) ensures that the dynamically adapted personalized information is factually consistent. We present four constraint satisfaction methods that cumulatively work to maximize collaboration and minimize conflicts between a set of information-snippets in order to dynamically generate personalized information.

1 Introduction

Constraints arise in most areas of human endeavor and we are used to solving them in an unambiguous and efficient manner. Computationally, constraint satisfaction methods allow the efficient navigation of large search spaces to find an optimal solution that entails the assignment of values to problem variables subject to given constraints [1,2]. Constraint satisfaction programming has been successfully applied to many problem areas that demand the hard search for a solution, such as *configuration* [3], *planning* [4], *resource allocation* [5] and *scheduling* [6], and lately many new and interesting applications of constraint satisfaction are emerging.

The profusion of web-based information resources hosting large volumes of diverse information content offers a mixed outlook to users. On the one hand, there is comfort in the fact that information is available for use if and when needed, yet on the other hand there is an apprehension considering the effort required to sift and process the available information in order to achieve a meaningful impact. *Information Personalization* (IP) research attempts to alleviate the cognitive overload experienced by users in processing and consuming generic, non-focused information content [7]. Put simply, IP involves the dynamic adaptation of generic information content to generate personalized information content that is intelligently designed to suit an individual's demographics, knowledge, skills, capabilities, interests, preferences, needs, goals, plans and/or usage behavior [8, 9]. To date, there are a number of web-mediated information services that provide personalized information for a variety of reasons, including healthcare [10], customer relationships [11], product promotions, education [12] and tourism. At the forefront of such IP initiatives are adaptive hypermedia systems [13] that manifest a hybrid of artificial intelligence methods—in particular natural language processing, case-based [14], model-based, and rule-based methods—to provide a variety of IP methods and perspectives [15].

In our work we investigate the modeling of IP as a constraint satisfaction problem. In our view, IP is achieved by selecting multiple highly-focused information-objects, where each information-object may correspond to some aspect of the user-model, and appending these user-specific information-objects to realize a seamless personalized information package. The process of IP, therefore, can be modeled as a constraint satisfaction problem that involves the satisfaction of two constraints: (1) given a large set of available information-objects, the constraint is to select only those information-objects that correspond to the user-model; and (b) given the selection of multiple user-compatible information-objects, the constraint is to retain only those information-objects that cumulatively present a factually consistent view—i.e. the contents of the retained information-items do not contradict each other.

In this paper, we present an intelligent constraint-based *information personalization framework* that (a) generates personalized information via the dynamic selection of multiple topic-specific information-objects deemed relevant to a user-model [8]; and (b) ensures that the dynamically adapted personalized information, comprising multiple topic-specific information-objects, is factually consistent. We present a unique hybrid of adaptive hypermedia and variations of existing constraint satisfaction methods that cumulatively work to maximize collaboration and minimize the conflicts between a set of information-objects to generate personalized information.

2 The Problem of Information Personalization

From an adaptive hypermedia perspective IP is achieved at three levels: (i) *Content adaptation* involves both linguistic changes to the information content and changes to the composition of text fragments that jointly make-up the finished personalized hypermedia document; (ii) *Structure adaptation* involves dynamic changes to the link structure between the hypermedia documents; and (iii) *Presentation adaptation* involves changes to the physical layout of content within the hypermedia document [9].

Content adaptation is the most interesting and challenging strategy for IP, because it involves the dynamic selection of multiple information-objects that correspond to a given user-model, and then their synthesis using a pre-defined document template to realize a personalized information. We argue that although existing IP methods generate highly focused personalized information vis-à-vis the user-model, they do not take into account the possibility that the ad hoc synthesis of heterogeneous information-objects (*albeit* the information-objects are relevant to the user) might unknowingly compromise the overall factual consistency of the personalized information content. Combining two information-objects can inadvertently lead to the generation of factually inconsistent information—i.e. one information-object stating a certain fact/recommendation whilst another information-object simultaneously contradicting the same fact/recommendation. We believe that in the absence of a content consistency checking mechanism, when multiple information-objects are synthesized, doubts may remain over the factual consistency of the personalized information.

Our definition of an IP problem therefore states that the scope of IP should not be limited to satisfying the user profile only, rather the IP strategy should also ensure that the personalized information content is factually consistent—i.e. no aspect of the personalized information content should be in contradiction with any other information simultaneously presented to the user. Hence, IP can be viewed as the satisfaction of two different constraints: (1) matching user-model attributes with information-object attributes to select user-specific information content; and (b) establishing information content consistency between multiple information-objects to ensure the factual consistency of the personalized information content.

2.1 Problem Specification

We approach the problem of IP at the content adaptation level. Our work is based on text fragment variants [11, 8], whereby a set of text fragments (or documents) are dynamically selected in accordance with the various aspects of a user profile. At runtime, the set of selected text fragments are systematically amalgamated to realize a hypermedia document containing personalized information. The problem of IP, from an optimization perspective, can therefore be specified as:

Given: (1) a *user-model* that comprises a number of user-defining attributes that describe the individual characteristics of a user; (2) a corpus of hypermedia documents called *Information Snippets* (IS). As the name suggests, each IS contains a text fragment of highly focused information that is pertinent to users with specific user-attributes. The IS are organized in a taxonomy that has four levels, as shown in Fig. 1.

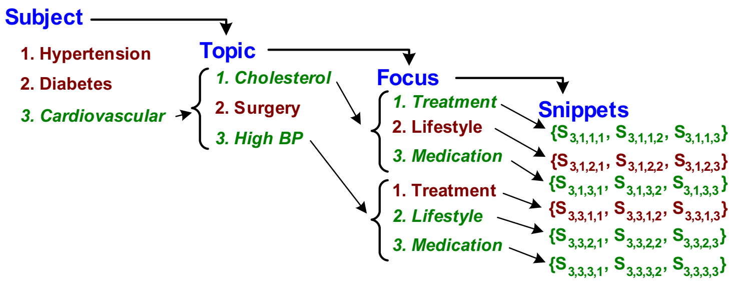

Subject
1. Hypertension
2. Diabetes
3. Cardiovascular
Topic
1. Cholesterol
2. Surgery
3. High BP
Focus
1. Treatment
2. Lifestyle
3. Medication
1. Treatment
2. Lifestyle
3. Medication
Snippets
$\{S_{3,1,1,1}, S_{3,1,1,2}, S_{3,1,1,3}\}$
$\{S_{3,1,2,1}, S_{3,1,2,2}, S_{3,1,2,3}\}$
$\{S_{3,1,3,1}, S_{3,1,3,2}, S_{3,1,3,3}\}$
$\{S_{3,3,1,1}, S_{3,3,1,2}, S_{3,3,1,3}\}$
$\{S_{3,3,2,1}, S_{3,3,2,2}, S_{3,3,2,3}\}$
$\{S_{3,3,3,1}, S_{3,3,3,2}, S_{3,3,3,3}\}$

Fig. 1. A taxonomy of information snippets. A traversal through the taxonomy is shown by following the italicized text from subject to topic to focus to snippets.

For an exemplar healthcare IP problem, at the highest level the *Subject* can be broadly classified into cardiovascular disease, diabetes, hypertension, etc. Each subject is further classified into *Topics*, for instance cardiovascular disease can be described in terms of cholesterol management, heart surgery, diagnostics, high BP etc. Each topic then entails multiple *Focus* areas, each focus area referring to a different aspect of the topic, for instance the different focus areas for cholesterol management are lifestyle, diet and medications. Finally, for each focus area there is a set of *Information Snippets*, where each IS contains information relevant to a specific focus area and targets specific user-attribute values such as age, gender, education level, etc.

Required: IP requires the automatic generation of the most *comprehensive*, *factually consistent* and *personalized* information package comprising a number of relevant IS that are systematically selected from the corpus and organized to yield a final personalized *Information Package.*

Constraints: The above three requirements translate into the following constraints: *Personalized*—the final information package should comprise all ISs that are consistent with the user-model; *Factual Consistency*—maintaining the personalized constraint, the final information package should ensure inter-IS consistency such that any two (or more) ISs should not give conflicting or inconsistent information; *Comprehensiveness*—maintaining the factual consistency constraint, the final information package should include the largest possible set of ISs that satisfy all constraints, and most importantly ensure that each focus area is minimally covered by a single IS.

Solution: The above problem specification brings to relief an interesting optimization problem, whereby the problem space on the one hand encompasses a wide diversity of users, whilst on the other hand a large volume of generic information content (in terms of ISs). The IP solution therefore involves searching the available ISs with respect to the user's characteristics, and selecting the largest possible set of relevant IS that jointly present a factually consistent view of the topic in question.

2.2 Operational Considerations

User-Model: A user-model comprises a set of user-defining attributes, each describing a particular characteristic of a user. Each user-attribute (UA) is represented as the tuple shown below:

$$\text{UA}(\textit{attribute, value, weight}) \quad 0 \le \textit{weight} \le 1,\ 0 \rightarrow \text{absent},\ 1 \rightarrow \text{present}$$

Where *attribute* refers to a user characteristics such as age, gender; *value* denotes the numeric or symbolic measurement of the attribute; and *weight* refers to the presence or absence of that particular attribute's value in the user-model. For example, UA(age, 40, 1) implies that the age of the user equaling 40 is valid. And, UA(allergy, pollen, 0) implies that the user does not have allergy to pollen.

Information Snippet (IS): An IS is represented in the form of a *conditional frame* that involves the binding of information content with a set of conditions [16]. Each IS is composed of two sections: (a) *Content section* that withholds the information content; and (b) *Condition section* that specifies the conditions for the selection of the document. The condition section comprises two types of conditions: (a) *Snippet-Selection Conditions* (SSC) that are compared with the user's model in order to determine whether the said IS is relevant to the user. An IS is selected if *all* SSC are satisfied; and (b) *Snippet-Compatibility Conditions (SCC)* determine whether the said

IS can mutually co-exist with other selected IS. An IS is selected if *all* SCC are satisfied. Both these conditions are representation by the tuple:

SSC/SCC (*context, value, weight*)

$0 \leq weight \leq 1$, $0 \rightarrow$ not recommended, $1 \rightarrow$ recommended

In the condition tuple, the context determines the nature of the condition, value states the text or numeric description of the condition, and *weight* defines the degree of the condition ranging from 0 to 1. For example SSC(*allergy, pollen, 0*) means the context of the condition pertains to allergies, the specific value of the context is pollen, and the weight being 0 implies not recommended. Hence, an IS with the above SSC cannot be selected for a user who has an allergy to pollen. Similarly, the SCC(*drug, aspirin, 0*) means the IS is compatible with all IS that *do not* recommend the drug named aspirin.

3 Modeling Information Personalization as a Constraint Satisfaction Problem

3.1 Constraint Satisfaction: An Overview

Mathematically speaking, constraints formalize the dependencies in a physical world in terms of a logical relation among several unknowns (or variables), each taking a value from a defined domain. In principle, a constraint restricts the possible values that the variables can take whilst solving a problem. *Constraint programming* solves problems by stating constraints about the problem area and consequently finding solutions that may 'satisfy' all the constraints. A *Constraint Satisfaction Problem* is defined by a tuple $P = (X, D, C)$ where $X=\{X_1, \dots , X_n\}$ is a finite set of *variables*, each associated with a domain of discrete values $D = \{D_1, \dots, D_n\}$, and a set of constraints $C = \{C_1, \dots, C_l\}$. Each constraint C_i is expressed by a relation R_i on some subset of variables. This subset of variables is called the *connection* of the constraint and denoted by $con(C_i)$. The relation R_i over the connection of a constraint C_i is defined by $R_i \subseteq D_{i1} \times \dots \times D_{ik}$ and denotes the tuples that satisfy C_i. A solution to a constraint satisfaction problem is an assignment of a value from its domain to every variable, in such a way that every constraint is satisfied [1, 2, 3]. This may involve finding (a) just one solution with no preferences, (b) all solutions, or (c) an optimal solution given some objective function defined in terms of some or all of the variables.

Solutions to a constraint satisfaction problem can be found by systematically searching through the possible assignments of values to variables using several different approaches. Popular approaches include the *Generate-and-Test* methods [17] that systematically generate each possible value assignment and then test to see if it satisfies all the constraints, and *Backtracking* methods [18] that incrementally attempt to extend a partial solution toward a complete solution. Both search methods guarantee a solution, if one exists, or else prove that the problem is insoluble [19].

Generate-and-Test methods generate all the possible solutions in the search space and then test each solution to determine whether it is the right solution. In doing so, each possible combination of the variable assignments is systematically generated and tested to see if it satisfies all the constraints. The first combination that satisfies all the constraints is taken as the solution. *B*acktracking search methods sequentially instantiate the variables in some order, and as soon as all the variables relevant to a constraint are instantiated, the validity of the constraint is checked. If the constraint is not satisfied, backtracking is performed to the most recently instantiated variable that still has alternative values available for examination. In this way, backtracking has the advantage of extending a partial solution that specifies consistent values for some of the variables towards a search for a complete solution [17, 18, 19]. Another approach for constraint satisfaction involves *Consistency* techniques that detect inconsistent values that cannot lead to a solution, and thus prune them from the search space to make the search more efficient [20, 21]. *Node Consistency* is the simplest consistency technique that works as follows: The node representing a variable V in a constraint graph is ***node consistent*** if for every value X in the current domain of V, each unary constraint on V is satisfied. If the domain D of a variable V contains a value Z that does not satisfy the unary constraint on V, then the instantiation of V to Z will always result in failure. This implies that node inconsistency can be eliminated by simply removing those values from the domain D of each variable V that do not satisfy the constraint on V.

3.2 Our CS-Based Information Personalization Approach

Given a subject and its constituent topics, we provide information personalization at the topic-level. For each topic in question, the search strategy is to select the most relevant and consistent IS for all its focus areas (see taxonomy shown in Fig. 1).

We define IP in a constraint satisfaction context as (a) a set of focus areas for a given topic, represented in terms of *focus-variables* $X=\{x_1,...,x_n\}$, where for each focus-variable x_i, there is a finite set of (focus-specific) IS. The set of IS associated with each focus-variable is deemed as its domain, D_i; (b) a user-model represented as a single-valued *user-variable*; and (c) and two types of constraints—*user-model constraint* and *co-existence constraint.* A solution to our constraint satisfaction problem is the systematic selection of the largest subset of IS associated with each topic—this is achieved by selecting the largest subset of IS for each focus-variable associated with the said topic—in such a way that the given user-model and co-existence constraints (amongst all selected IS) are fully satisfied. Such a constraint satisfaction solution can be obtained by searching the domain for each focus-variable. Our constraint satisfaction approach for searching the solution is given as follows:

Step 1-Selection of user-specific information content: The user-model attributes forms the basis for selecting user-specific IS. Node-consistency based techniques are used to solve the *user-model constraint* by satisfying the snippet-selection conditions of each IS (where the IS is related to the given topic by a focus variable) with the user-attributes noted in the user-model. We collect a *candidate-IS set* that comprises all possible (topic-specific) ISs that are relevant to the user-model (shown in Fig. 2b).

Step 2-Selection of 'Core' information content: Given the candidate-IS set, it is important to ensure that the selected ISs can potentially co-exist with each other without causing any factual inconsistency. Hence the next step is to establish the minimum information coverage that is factually consistent—i.e. establishing the *core-IS set* which includes a single IS for each focus area in question. We use backtracking search to satisfy the *co-existence constraints* by globally satisfying the snippet-compatibility conditions for all the IS in the candidate-IS set. Any IS that is deemed factually inconsistent with the rest of the IS is discarded. The resulting *core-IS set* (as illustrated in Fig. 2c) depicts the minimum coverage of factually consistent information whilst also satisfying the requirement for *comprehensiveness*—i.e. to minimally cover each focus area with a single IS for all topics in question.

The rationale for generating a core-IS set is to initially establish a baseline of factually-consistent ISs that meet the comprehensiveness requirement. The core-IS set provides limited information coverage, but more importantly the information is factually consistent—our thinking being that it is better to give less information but ensure that it is consistent, than to give more information that maybe potentially inconsistent. Having established a baseline (or minimum) factually consistent information, in the next steps we attempt to build on the core-IS set to extend the information coverage.

Step 3-Selection of 'Extended' information content: Given the core-IS set, we next attempt to maximize its information coverage by including previously non-selected candidate-ISs (in step 2) to the core-IS set, whilst ensuring that the overall factual consistency is maintained. We use the stochastic generate-and-test method to 'stochastically' search for previously non-selected candidate-ISs that satisfy the co-existence constraint with the core-IS set. If the co-existence constraint is satisfied, the candidate-IS is included to the core-IS set resulting in an *extended-core-IS set* which will then be used as the baseline for future inclusions of other candidate-ISs. Note that if no additional candidate-IS can be included to the core-IS set then the extended-core-IS set equals the core-IS set. The outcome of this step is a more optimal extended-core-IS set that represents the new, yet potentially larger than before, minimum information coverage that satisfies both the user-model and co-existence constraints (shown in Fig. 2d). In the next step we attempt to maximize the information coverage.

Step 4-Selection of 'Optimal' information content: The generation of the core-IS set and the follow-up extended-core-IS set involved the use of stochastic search algorithms that were solely designed to satisfy the co-existence constraints between the candidate-IS, without checking the possibility that the selected candidate-IS may in turn block the future inclusion of other candidate-IS to the core- and extended-core-IS sets. It is fair to assume that due to the stochastic nature of the solution, there may exist the possibility that a particular candidate-IS may satisfy the prevailing co-existence constraint situation at that time and become a member of the core- or extended-core-IS set, but being inconsistent with a large number of non-selected candidate-ISs it may block their potential inclusion to the extended-core-IS set, thus contributing to a sub-optimal solution. Having said that, the exclusion of a single sub-optimal candidate-IS from the extended-core-IS set may enable the potential inclusion of multiple non-selected candidate-ISs to the extended-core-IS set, whilst still maintaining the co-existence constraints and the comprehensiveness requirement.

In order to further optimize the information coverage, our approach is to explore the possibility of replacing a single sub-optimal IS in the extended-core-IS set with multiple non-selected candidate-IS. This is achieved by our novel information optimization mechanism, termed as *snippet swapping*. The snippet swapping mechanism generates the most optimal information coverage in terms of the final *presentation-IS set* (shown in Fig. 2e), that (a) maintains the co-existence constraints, and (c) ensures that each focus area (for all selected topics) is represented by at least one IS. Note that if snippet swapping is not possible then the presentation-IS set equals the extended-core-IS set. In conclusion, the optimized presentation-IS set is the final CSP solution.

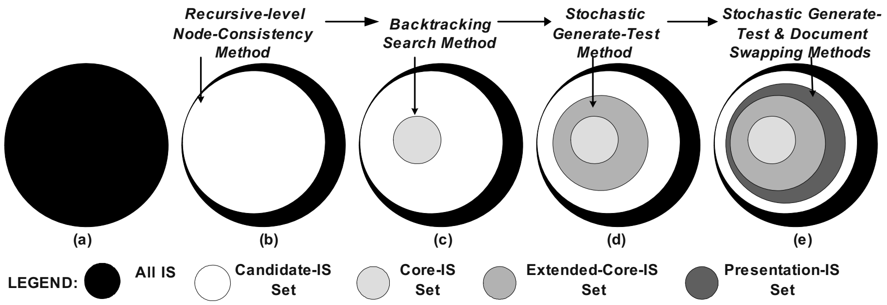

Recursive-level Node-Consistency Method
Backtracking Search Method
Stochastic Generate-Test Method
Stochastic Generate-Test & Document Swapping Methods
(a)
(b)
(c)
(d)
(e)
LEGEND:
All IS
Candidate-IS Set
Core-IS Set
Extended-Core-IS Set
Presentation-IS Set

Fig. 2. Schematic representation of the different stages of the CSP solution, highlighting the respective maximization of the information coverage at each progressive stage.

4 Constraint Satisfaction Methods for Information Personalization

In line with the abovementioned IP approach we have developed variants of consistency-checking techniques and search algorithms to generate the personalized presentation-IS set. In the forthcoming discussion we present our variants of constraint satisfaction methods that are used to solve the user-model constraint to generate the candidate-IS set, and the co-existence constraints to generate the core- and extended-core IS sets, and the snippet-swapping method to generate the presentation-IS set.

4.1 User-Model Constraint Satisfaction: Generating the Candidate-IS Set

A user-model constraint between a *focus-variable* and a *user-variable* is satisfied when all the IS in the domain of the focus-variable are consistent with the user-model. The general idea is to compare the snippet-selection conditions (SSC) for each IS with the user-attributes (UA) listed in the user-model (UM) as follows, $(context, value)_{SSC}^{IS} = (attribute, value)_{UA}^{UM}$. We calculate a *conflict value* (CV), as shown below, between the SSC and UA to determine constraint satisfaction. A low CV value implies that the user-model constraint has been satisfied and that the IS is deemed relevant to the user, whereas a high CV value denotes the irrelevance of the IS to the user. The acceptance level of CV is a parameter that can be set the user to determine the desired severity of the SSC. The CV is the modulus of the difference between the weights of the SSC and the matching UA, and is calculated as follows:

$$CV_{SSC}^{UA} = \left|(weight)_{SSC}^{IS} - (weight)_{SSC}^{UM}\right|,\ (context, value)_{SSC}^{IS} = (attribute, value)_{UA}^{UM}$$

$$0 \le CV_{SSC}^{UA} \le 1\ ;\ where\, 0 \rightarrow const.\ satisfied\, ,\ 1 \rightarrow const. not\ satisfied$$

To satisfy the user-model constraint we employ a variation of CSP node-consistency technique—the *recursive-level node-consistency algorithm* [2]. The working of our modified recursive-level node-consistency algorithm is as follows: for each focus-variable, if the domain contains an IS that is inconsistent towards the user-model, then that particular IS is removed from the domain. Eventually, only those IS that are consistent with the user-model are retained in each focus-variable's domain and the resulting set of user-specific IS are regarded as the *candidate-IS set*.

```
Algorithm Recursive-level Node Consistency
  for focus-var$_1$ to focus-var$_m${m = number of focus areas}
     for IS$_1$ to IS$_n$ {n = no. of IS in the domain of focus-var$_i$}
       test UMC  {UMC = user model constraint}
       if UMC not satisfied {inconsistent with user-model}
             discard IS$_i$
       endif
     endfor
  endfor
```

4.2 Co-existence Constraint Satisfaction I: Generating the Core-IS Set

Co-existence constraints between two focus-variables need to be satisfied to ensure that their respective selected ISs are factually consistent with each other. In practice, co-existence constraints between two focus-variables$_{A\&B}$ are satisfied if the selected ISs from the domain of focus-variable$_A$ are consistent with the selected ISs from the domain of focus-variable$_B$. Two SCC are only comparable if they both have the same content and value, as follows: $(context, value)_{SCC_A}^{IS_A} = (context, value)_{SCC_B}^{IS_B}$. The SCC of an IS is satisfied with respect to the SCC of another IS. A co-existence constraint is not-satisfied when the conflict value (CV) exceeds a predefined user threshold.

$$CV_{SCC_A}^{SCC_B} = \left|(weight)_{SCC_A}^{IS_A} - (weight)_{SCC_B}^{IS_B}\right|,\ (context, value)_{SCC_A}^{IS_A} = (context, value)_{SCC_B}^{IS_B}$$

$$0 \le CV_{SCC_A}^{SCC_B} \le 1\ ;\ where\, 0 \rightarrow const.\ satisfied\, ,\ 1 \rightarrow const. not\ satisfied$$

To satisfy co-existence constraints leading to the generation of the core-IS set we employ a Backtracking (BT) search method. The BT method searches the candidate-IS space to generate the core-IS set by (i) choosing an un-instantiated focus-variable, i.e. no IS has yet been assigned to the focus-variable; (ii) choosing a candidate-IS from the domain of the un-instantiated focus-variable; (iii) checking whether the candidate-IS is consistent with ISs that have already been selected to instantiate the other focus-variables; (iv) if the candidate-IS is consistent—implying that the co-existence constraint is satisfied—it is selected by instantiating the focus-variable, else the next candidate-IS within the domain of the same focus-variable is examined. Given that the co-existence constraint cannot be satisfied because all the candidate documents for a focus-variable have been checked, backtracking is performed to select the most recently instantiated focus-variable that may still have some alternative candidate-ISs and then search forward again based on the new instantiation of the said focus-variable. Successful BT search ensures that each focus-variable is instantiated with an IS, thus satisfying the minimum comprehensiveness requirement, and resulting in the *core-IS set*. The order in which the topics are searched can be based on the following schemes: (1) Original chronological order of the topics; (2) Randomly selecting the next topic to search.; (3) User-specified search order of the topics; important topics are search first followed by the less significant topics; (4) Partial user-specified order (the starting topic and maybe a few others are given) and the remaining topics are selected in a random order.

4.3 Co-existence Constraint Satisfaction II: Generating the Extended-Core-IS Set

Extension of the core-IS set to the potentially larger extended-core-IS set is performed via the *Stochastic Generate and Test* (S-GT) method. The motivation for generating the extended-core-IS set is to maximize the current information coverage by selecting previously non-selected candidate-IS that do not violate the a priori established factual consistency of the core-IS set.

The working of the S-GT method is as follows: the non-selected candidate-IS are randomly sequenced in N different groups. Each group of ISs is then systematically searched based on the sequence of the constituent ISs in the group in an attempt to include more candidate-IS into the core-IS set without violating the co-existence constraint. Consequently, N extended-core-IS sets are generated, whereby the extended-core-IS set with the most ISs is selected. We argue that the S-GT method is suitable for this purpose because of its stochastic nature in selecting focus-variables and evaluating the ISs within their domain in a manner that avoids the ‘unfair’ effects resulting from a sequenced evaluation of ISs as practised by most search algorithms.

4.4 Snippet Swapping: Generating the Presentation-IS Set

The information coverage of the extended-core-IS set can be further increased by including more non-selected candidate-IC, but at this stage this is only possible by removing an IS in the extended-core-IS set. The basic idea is to ‘swap’ a single IS in the extended-core-IS set with multiple candidate-ISs—this is reflective of the situation when a single IS in the extended-core-IS set is factually inconsistent with multiple candidate-IS, hence it is single-handedly blocking the inclusion of multiple candidate-ISs to the extended-core-IS set. The snippet swapping algorithm, given below, explains the thinking behind the snippet swapping mechanism.

Algorithm Snippet Swapping
```
for each $IS_A$ in the extended-core-IS set
    identify the non-selected candidate-ISs that are
    inconsistent to $IS_A$
    if size of non-selected candidate-ISs $N$ > 1
        if $IS_A$ is not the only IS selected for a focus-variable
```

```
          apply S-GT algorithm to the non-selected candi-
          date-ISs to generate N sets
          if size of the largest set of candidate-IS C > 1
               discard IS_A
               append C to the extended-core-IS set
          endif
        endif
     endif
endfor
```

The snippet swapping mechanism extends the information coverage whilst still maintaining the factual consistency and comprehensiveness requirements of the result presentation-IS set.

5 Generating Personalized Healthcare Information

We present a working example of constraint satisfaction based IP as per our approach discussed earlier. The scenario involves a person suffering from two health problems—i.e. high BP and arthritis—and we need to provide personalized healthcare information based on his user-model given in Table 1.

Table 1. An exemplar user-model

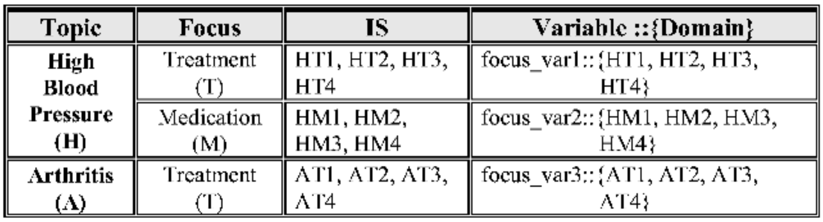
Health Problemss					
1. High Blood Pressure		2. Arthritis			
User Attributes (UA)					
Attribute	**Value**	**Weight**	**Attribute**	**Value**	**Weight**
Age	45	1	Medication	DrugY	1
Gender	Male	1	Lifestyle	Smoker	1
Education	Graduate	1	Lifestyle	Active	0
Family History	Diabetes	0	Allergy	Pets	1
Medication	DrugX	0	Allergy	Pollen	0

As per our IS organization taxonomy (given in Fig. 1), the two topics are *high BP* and *arthritis*, each having two focus areas namely *treatment* and *medication*. Table 2 illustrates the set of IS available for each focus area for each topic. We need to define the focus-variable (*focus_var*) representing each focus area, such that the domain for each focus-variable comprises the ISs that correspond to the focus. Due to space limitations we will not be able to show the processing for each focus variable, however for illustration purposes the outcome of the CS methods for focus_var1 are shown.

Step 1- Generate Candidate-IS set: This involves the satisfaction of user-model constraints using the node-consistency algorithm. Table 3 shows the candidate-IS set for focus_var1, whereby only the ISs that are relevant to the user-model are selected.

Step 2- Generate Core-IS set: This step involves the satisfaction of the co-existence constraints for each IS (not be shown due to lack of space). Table 4 shows the core-IS set derived from the candidate-IS set for each focus variable. Note that the core-IS set comprises the first ISs in the focus_var list —i.e. HT1, HM1 and AM1— for the focus_var1 (HT), focus_var2 (HM) and focus_var4 (AM). This is because the or the focus_var1 (HT), focus_var2 (HM) and focus_var4 (AM). This is because the search algorithm starts with the first IS in the focus_var list. Interestingly enough, for the focus_var3 the third IS—i.e. AT3—is selected because firstly AT1 was in conflict with HM1 and then secondly AT2 was in conflict with HT1. Since, both HM1 and HT1 were already a member of the core-IS set when the evaluation for AM was concluded, hence an IS was chosen that could co-exist with the a priori members of the developing core-IS set. The affect of sequencing of IS for evaluation and subsequent selection is addressed in the snippet-swapping stage.

Table 2. IS for the topics high blood pressure and arthritis. Also shown is the defintion of variables (focus_var) for each focus area in the realm of a topic.



Topic	Focus	IS	Variable ::{Domain}
High Blood Pressure (H)	Treatment (T)	HT1, HT2, HT3, HT4	focus_var1::{HT1, HT2, HT3, HT4}
	Medication (M)	HM1, HM2, HM3, HM4	focus_var2::{HM1, HM2, HM3, HM4}
Arthritis (A)	Treatment (T)	AT1, AT2, AT3, AT4	focus_var3::{AT1, AT2, AT3, AT4}

Table 3. The candidate-IS set for the topic high blood pressure and focus area is treatment.

Doc	Snippet Selection Condition	Matching User Attribute	CV	Status
HT1	<gender, male, 1>	<gender, male, 1>	0	**Retained**
HT2	<family history, diabetes, 1>	<family history, diabetes, 0>	1	Discarded
HT3	<medication, DrugX, 1>	< medication, DrugX, 0>	0	**Retained**
HT4	< allergy, seafood, 1>	<condition, pregnant, 0>	1	Discarded

Step 3 – Generate the Extended-Core-IS set: Next, we attempt to increase the information coverage of the core-IS set by applying the stochastic generate and test algorithm as per our approach for generating the extended-core-IS set. Table 5 shows the three random sets of non-selected candidate-IS, whereby the third random set is shown to best maximize the information coverage.

Note the stochastic nature of the search as the random ordering of the ISs for SCC satisfaction affects the outcome. In the third set, AT4 which is inconsistent with both HM4 and HT3 was positioned after HM4 in the random set. This enabled HM4 to be selected first instead of AT4 and thus blocked AT4 to be selected subsequently. Without AT4 in the extended-core-IS set it was possible for HT3 to be next selected. Note that this situation was not possible for the first two random sets. AM2 was not selected because it was in conflict with HM1 which was a member of the core-IS set. AT1 and AT2 are still non-selectable as they conflict with two members of the core-IS set.

Step 4- Generate the Presentation-IS set: Finally, we attempt to achieve optimal information coverage by applying the snippet swapping mechanism to generate the presentation-IS set. Table 6 shows the extended-core-IS set (comprising 8 ISs) together with their conflicts with IS discarded during BT and S-GT search. HM1 has been detected to be blocking two candidate-ISs—i.e. AT1 and AM2. Since the *Topic:Focus* area for High *BP:Medication* is represented by both HM3 and HM4 in the extended-core-IS set, it is possible to swap HM1 with AT1 and AM2 without disturbing the factual consistency and still maintaining the completeness requirement. As a result we get an optimal presentation-IS set (as shown in Table 7) that is larger than the initial extended-core-IS set. The resultant presentation-IS set is the solution of the IP problem and represents the personalized and factually consistent information suited for a specific user-model.

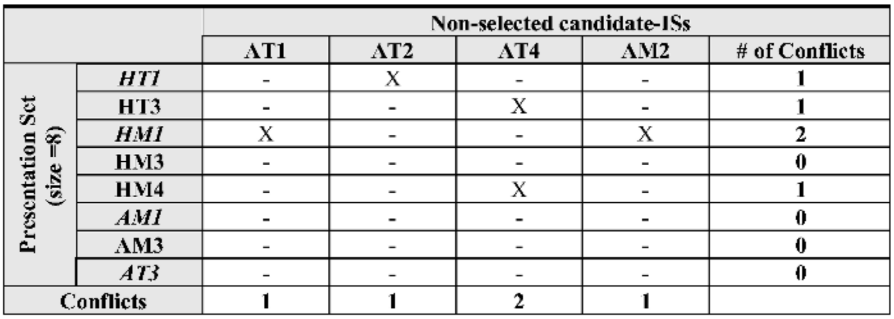
Table 4. Given the candidate-IS set (coverig all focus areas), we illustrate the core-IS set derived using backtracking search method. Also shown are the non-selected candidate-IS.

Topic-variables	Domain (Candidate-IS set)	Core-IS set	Non-selected Cand.-IS
focus_var1	HT1, HT3	**HT1**	HT3
focus_var2	HM1, HM3, HM4	**HM1**	HM3, HM4
focus_var3	AT1, AT2, AT3, AT4	**AT3**	AT1, AT2, AT4
focus_var4	AM1, AM2, AM3	**AM1**	AM2, AM3

Table 5. An extended-core-IS set resulting from S-GT search over three random sets of IS

Non-selected candidate-IS arranged in a random order	IS selected	IS discarded	Size
AT4, HM3, AM3, AT1, HM4, AM2, HT3, AT2	AT4, HM3, HT3	AM3, AT1, HM4, AM2, AT2	3
AM2, AT1, AT4, HM4, HT3, AT2, HM3, AM3	AT4, HT3, HM3	AM2, AT1, HM4, AT2, AM3	3
HM4, AT4,AT2, HT3, HM3, AM2, AT1, AM3	**HM4, HT3, HM3, AM3**	**AT4, AT2, AM2, AT1**	**4**

Table 6. Extended-core-IS set *before* optimization. The italized IS are members of the core-IS set, whereas the others were added later during the extended-core-IS generation step.



		Non-selected candidate-ISs				
		AT1	**AT2**	**AT4**	**AM2**	**# of Conflicts**
Presentation Set (size =8)	***HT1***	-	X	-	-	**1**
	HT3	-	-	X	-	**1**
	HM1	X	-	-	X	**2**
	HM3	-	-	-	-	**0**
	HM4	-	-	X	-	**1**
	AM1	-	-	-	-	**0**
	AM3	-	-	-	-	**0**
	AT3	-	-	-	-	**0**
Conflicts		**1**	**1**	**2**	**1**	

Table 7. An optimal presentation-IS set *after* optimization.

		Non-selected ISs			**# of Conflicts**
		AT2	**AT4**	**HM1**	
Presentation Set (Size = 9)	***HT1***	X	-	-	**1**
	HT3	-	X	-	**1**
	HM3	-	-	-	**0**
	HM4	-	X	-	**1**
	AM1	-	-	-	**0**
	AM3	-	-	-	**0**
	AT3	-	-	-	**0**
	AT1	-	-	X	**1**
	AM2	-	-	X	**1**
Conflicts		**1**	**2**	**2**	

5.1 Evaluation

The evaluation of the featured IP method focused on establishing the completeness and factual consistency of the information package. The computational complexity of the search methods were not measured as it was not deemed to be the most pressing issue at this stage, however we will present the computational complexity of the various methods in a separate publication. We anticipated that the random nature of the CS search methods might have a significant bearing on the final output, because the manner in which the initial IS are selected determines the overall makeup of the final output. For that matter, the document swapping method introduced here provides an opportunity to re-visit the selected IS (extended core-IS set) and to optimize the presentation set. The experimental data comprised: 10 topics each with 2 focus areas; 70 IS each with constraints; and 10 controlled user-models. Given the experimental data, experiments were carried out to evaluate the completeness and consistency of the final output. Analysis of the output—i.e. the personalized information package—indicated that whenever the completeness criteria was satisfied all the IS present in the presentation set were found to be consistent with each other. This observation vindicates the efficacy of the CS methods deployed to achieve IP.

6 Concluding Remarks

Person-specific customization of information viz. a user-model is a complex task that necessitates a systematic, pragmatic and multifaceted strategy. In this paper we presented and demonstrated an IP framework that purports a unique hybrid of adaptive hypermedia and constraint satisfaction methods. We have demonstrated the successful application of constraint satisfaction methods for information personalization that offers an alternate and interesting perspective to research in both information personalization and application of constraint satisfaction methods.

In conclusion, we believe that this is the first step towards the incorporation of constraint satisfaction within an information personalization paradigm. Also, the realization to ensure factual consistency when amalgamating heterogeneous information will lead to interesting research in adaptive information delivery systems. Finally, we believe that the featured IP approach can be used for a variety of E-services for education material customization, stock market reporting and advice, tourist information and so on; the only limitation is the specification of co-existence constraints which demands human expert involvement—a likely bottleneck.

References

[1] Tsang E, *Foundations of constraint satisfaction*. Academic Press, London, UK. 1993.

[2] Barták R, Constraint programming: In pursuit of the holy grail. *Proceedings of the Week of Doctoral Students (WDS99),* Part IV, MatFyzPress, Prague, 1999, pp. 555-564.

[3] Sabin D, Freuder E, Configuration as composite constraint satisfaction. *Proceedings of the Artificial Intelligence and Manufacturing Research Planning Workshop*, 1996, pp.153-161.

[4] Stefik M, Planning with constraints (MOLGEN: Part 1). *Artificial Intelligence*, Vol. 16(2), 1981, pp. 111-140.

[5] Sathi A, Fox MS, Constraint-directed negotiation of resource allocations. In *L. Gasser & M. Huhns (Eds.) Distributed Artificial Intelligence Volume II,* Morgan Kaufmann, San Mateo, 1989, pp. 163-194.

[6] Fox M, *Constraint-Directed Search: A Case Study of Job-Shop Scheduling*. Morgan Kaufmann, London, 1987.

[7] Zirpins C, Weinreich H, Bartelt A, Lamersdorf W, Advanced concepts for next generation portals. *Proceedings of 12th International Workshop on Database and Expert Systems Applications*, 3-7 Sept. 2001, pp. 501 –506.

[8] Fink J, Kobsa A, Putting personalization into practice. *Communications of the ACM*, 2002, Vol. 45(5).

[9] Perkowitz, M, Etzioni O, Adaptive web sites. *Communications of the ACM*, 2000, Vol. 43(8), pp. 152-158.

[10] Abidi SSR, Chong Y, Abidi SR, Patient empowerment via 'pushed' delivery of personalized healthcare Educational content over the internet. *Proceedings of 10th World Congress on Medical Informatics*, 2001, London.

[11] Kobsa A, Personalized hypermedia presentation techniques for improving online customer relationships. *Knowledge Engineering Review,* Vol. 16(2), 1999, pp. 111-155.

[12] Henze N, Nejdl W, Extendible adaptive hypermedia courseware: integrating different courses and web material. In *P. Brusilovsky, O Stock & C Strappavara (Eds.) Adaptive Hypermedia and Adaptive Web-based Systems*, Springer Verlag, 2000, pp. 109-120.

[13] Brusilovsky P, Kobsa A, Vassileva J (Eds.), *Adaptive Hypertext and Hypermedia*. Kluwer Academic Publishers, Dordrecht, 1998b.

[14] Abidi SSR, Designing Adaptive Hypermedia for Internet Portals: A Personalization Strategy Featuring Case Base Reasoning With Compositional Adaptation. In FJ Garijo, JC Riquelme & M Toro (Eds.) *Lecture Notes in Artificial Intelligence 2527: Advances in Artificial Intelligence (IBERAMIA 2002)*. Springer Verlag: Berlin, 2002. pp. 60-69.

[15] Zhang Y, Im I, Recommender systems: A framework and research issues. *Proceedings of American Conference on Information Systems*, 2002.

[16] Boyle C, Encarnacion AO, MetaDoc: An adaptive hypertext reading system. *User Models and User Adapted Interaction,* Vol. 4(1), 1994, pp. 1-19.

[17] Dechter R, Pearl J, Network based heuristics for constraint satisfaction problems. *Search in Artificial Intelligence,* Springer Verlag, Berlin, 1988, pp. 370-425.

[18] Kumar V, Algorithms for constraint satisfaction problems: A survey. *AI Magazine*, 1992, Vol. 13(1), pp. 3-44.

[19] Grzegorz K, van Beek, P, A theoretical evaluation of selected backtracking algorithms. *Artificial Intelligence,* 1997, Vol. 89, pp. 365-387.

[20] Mackworth AK, Consistency in networks of relations. Artificial Intelligence, 1977, Vol. 8, pp. 99-118.

[21] Freuder E, A sufficient condition for backtrack-free search. *Communications of the ACM*, 1982, Vol. 29(1), pp. 24-32.

On Selection Strategies for the DPLL Algorithm

Morten Irgens[1] and William S. Havens[1,2]

[1] Actenum Corporation
[2] Simon Fraser University

Abstract. This paper discusses selection strategies for constructive search algorithms. Existing selection strategies have been based on a belief that it is better to search where solutions are more likely to be, or that it is better to search where the search space is smallest. While we give evidence for both strategies, we also show that they are likely to give modest results. We introduce the utility strategy, show that utility heuristics are superior, and that there is an optimal utility balance. We have focused this work on how the Davis-Putnam-Logeman-Loveland algorithm ("DPLL") uses heuristics in solving satisfiability problems. Empirical results are given to justify our conclusions.

1 Introduction

A number of branching heuristics have been proposed for satisfiability (*sat*) and constraint satisfaction problems (*csp*). In this paper we want to identify the often unstated search strategies underlying these heuristics, and challenge some commonly held assumptions. For instance, it is not necessarily given that the best place to search is where the probability for finding a solution is highest. Neither is it necessarily given that the best place to search is where you get failures early. Assume the following hypotheses and their corresponding selection strategies:

- *'Does Not Matter':* Hypothesis: Branching rules do not have any significant performance difference. Strategy: Select randomly.
- *'Satisfaction':* Hypothesis: Other things being equal, a branching rule performs better when it creates subproblems that are more likely to be satisfiable [12]. Strategy: Select the subtree with the highest associated solution probability.
- *'Simplification':* Hypothesis: Other things being equal, a branching rule performs better when it creates subproblems with smaller search spaces. Strategy: Select the smallest subtree.
- *'Refutation':* Hypothesis: Other things being equal, a branching rule performs better when it creates subproblems that are less likely to be satisfiable. Strategy: Select the subtree with the smallest solution probability.

The main result of this paper is a refined understanding of how to create branching rules and why existing rules behave as they do. We investigate the strategies above and their associated hypotheses empirically, and we discuss why they behave as they do. While we support both the satisfaction and simplification strategy, we also establish that they have very little merit. We present a new strategy, which we call the utility strategy, and we show that it is superior to the other. We argue that the performance of many good heuristics today can be explained as instantiations of this strategy.

2 The Satisfiability Problem

We assume familiarity with the satisfiability problem, also known as *sat.* This paper will be concerned with k-sat, a restriction where all clauses are of equal length k which is NP-complete for all $k > 2$ [1]. We are given a universe of discourse $\texttt{BOOL} = \{\mathbf{t}, \mathbf{f}\}$, a set X of undefined symbols called atoms, $X = \{x_1 \dots x_n\}$, and set of connectives $\{\neg, \wedge, \vee\}$, which are predefined logical operators with the intended semantics of *and, or* and *not*, respectively. A *well formed formula* $\mathcal{F}$ is defined as any string that can be generated using the following grammar:

$$\mathcal{F} \longrightarrow \mathcal{F} \wedge \mathcal{F} \mid c, \qquad c \longrightarrow c \vee c \mid l, \qquad l \longrightarrow a \mid \neg a, \qquad a \longrightarrow x_1 \mid x_2 \mid \dots .$$

We see that formulae generated in this manner will be on the form of conjunctions of clauses, which again are disjunctions of atoms or their negations. The set of well-formed formulae is denoted $\mathcal{H}$. We are also given a *configuration function* $\tau : X \rightarrow \texttt{BOOL}$ which defines the truth values of the atoms, and an *evaluation function* $\gamma : \mathcal{H} \rightarrow \texttt{BOOL}$, which establishes the properties of the connectives.

A configuration function *assigns* truth values to the literals (which also is called to *configure*, *fix* or *interpret* them). We say that a configuration τ *satisfies* a formula if the formula evaluates to $\mathbf{t}$ under τ. A satisfying configuration of a formula is usually called a *solution* or a *model* of the formula. The variation of the satisfiability problem that we are interested in can be stated as: *Given a propositional formula on conjunctive normal form, find a model if one exists.*

The DPLL algorithm for solving the decision version of sat was presented Davis, Logemann and Loveland in 1962 [4] based on earlier work by Davis and Putnam [5]. It lends itself naturally to finding models for formulae. Figure 1 shows a version of the DPLL algorithm as a recursive chronological backtrack algorithm [17; 12]. In each iteration we pick a variable and guess which truth value it should have in the model. This is done by picking l from a set of literals $l_1, \dots l_k$, creating two new formulae, $\mathcal{F} \wedge l$ and $\mathcal{F} \wedge \neg l$, and recursing on the formula we believe is consistent. If we recurse on $\mathcal{F} \wedge l$, we say that we are *fixing* l. The algorithm propagates the consequences of fixing l, a process that can be seen as a function π on the formula $\mathcal{F} \wedge l$ returning a simplified formula $\pi(\mathcal{F} \wedge l)$. In DPLL, π is a combination of monotone variable fixing and unit resolution. Since these deductions simplify the formula by both removing and shortening clauses, repeatedly doing this will eventually result in either $\mathcal{F}$ becoming empty, or in the detection of an inconsistency. If the formula $\mathcal{F} \wedge l$ turns out to be inconsistent, we backtrack on $\mathcal{F} \wedge \neg l$ instead. Each time that we fix a literal, it is added as a unit clause to the set of clauses $\mathcal{F}$, and unit-resolve moves it to a set $\mathcal{A}$. In a successful run, $\mathcal{A}$ will contain the model.

```
procedure unit-resolve($\mathcal{F}$,$\mathcal{A}$);
  while $\mathcal{F}$ contains a unit clause $l$ do
    Add $l$ to $\mathcal{A}$;
    Remove from $\mathcal{F}$ all clauses containing $l$;
    Remove from $\mathcal{F}$ all occurrences of $\neg l$;
    if $\neg l$ is removed from a unit clause
      then report contradiction;
  endwhile;
end unit-resolution;

procedure monotone-variable-fixing($\mathcal{F}$,$\mathcal{A}$);
  while $\mathcal{F}$ contains a monotone literal $l$ do ;
    Remove from $\mathcal{F}$ all clauses containing $l$;
    Add $l$ to $\mathcal{A}$;
  endwhile;
end monotone-variable-fixing;
```

```
procedure DPLL($\mathcal{F}$,$\mathcal{A}$);
  unit-resolve($\mathcal{F}$,$\mathcal{A}$);
  monotone-variable-fixing($\mathcal{F}$,$\mathcal{A}$);
  if contradiction then return ;
  if $\mathcal{F}$ is empty then halt and output $\mathcal{A}$;
  else
    Pick a literal $l$ in $\mathcal{F}$;
    DPLL($\mathcal{F} \wedge l$, $\mathcal{A}$);
    DPLL($\mathcal{F} \wedge \neg l$, $\mathcal{A}$);
  endif
end DPLL;
```

Fig. 1. The DPLL algorithm and its supporting propagation algorithms.

3 Heuristics

The DPLL algorithm does not specify *how* to select which literal to fix. Strategies for literal selection are often called *branching* strategies. The efficiency of search algorithms like the DPLL algorithm is dependent on their branching strategies. In the following, we will investigate the strategies presented in the introduction by investigating some representative heuristics. The heuristics are all based on computationally inexpensive estimates of the effects of the propagation (rather than some type of lookahead).

3.1 Satisfaction Heuristics

Let $\mathcal{F}_l$ denote the sub-formula of $\mathcal{F}$ consisting of the set of clauses containing l, and $\mathcal{F}_{\widetilde{l}}$ the set of clauses that do not contain l. In the following, we assume these definitions:

$$J(\mathcal{F}) = \sum_{\{c \in \mathcal{F}\}} \frac{1}{2^{|c|}}, \qquad J(\mathcal{F}_l) = J(l) = \sum_{\{c \in \mathcal{F} | l \in c\}} \frac{1}{2^{|c|}}. \tag{1}$$

A first step towards computationally inexpensive estimates of the effects of propagation, is to note that we can divide the set of clauses into three subsets, those clauses containing l, those containing $\neg l$, and those containing neither, i.e. $\mathcal{F} = \mathcal{F}_l \cup \mathcal{F}_{\neg l} \cup \mathcal{F}_{\widetilde{l,\neg l}}$. The simplified formula $\pi(\mathcal{F} \wedge l)$ contains no clauses where l is a member (since those clauses get removed when l is fixed to true) and we have that $\pi(\mathcal{F} \wedge l) = \pi(\mathcal{F} \wedge l)_{\neg l} \wedge \mathcal{F}_{\widetilde{l \neg l}}$. Furthermore, clauses in $\mathcal{F}$ where $\neg l$ is a member get shortened, since the literal $\neg l$ gets removed. This means that when we fix a literal and unit-resolve, we have $J(\pi(\mathcal{F} \wedge l)_{\neg l}) = 2 \cdot J(\mathcal{F})$, which gives us the following theorem and heuristic.

Theorem 1. *Given the formulae $\pi(\mathcal{F} \wedge l_1)$ and $\pi(\mathcal{F} \wedge l_2)$ where π includes unit propagation, we have $P[\pi(\mathcal{F} \wedge l_1)$ is solvable$] \leq P[\pi(\mathcal{F} \wedge l_2)$ is solvable$] \Leftrightarrow J(\neg l_1) - J(l_1) \geq J(\neg l_2) - J(l_2)$.*

The first order probability heuristic *(fop) [12]: Choose the literal l with the highest value of $J(l) - J(\neg l)$, breaking ties randomly.*

For proof, see [12]. If we ignore either the clause lengths or the number of clauses satisfied, we get the following two heuristics:
The Jeroslow-Wang heuristic [15]: *(jw): Select the literal l with the highest value of $J(l)$, breaking ties randomly.*

The Reverse Jeroslow-Wang heuristic [12] *(rjw): Select the literal l with the highest value of $J(\neg l)$, breaking ties randomly.*

The jw heuristic was first presented by Jeroslow and Wang with the motivation that it selects “in an intuitive sense a system most likely to be satisfiable” [15, page 172]. The argument is that the heuristic satisfies and removes the subset of clauses with the smallest solution probability. It follows that the remaining literals will have clause subsets with higher solution probabilities and that it will leave us with the formula with the highest satisfaction probability.

The rjw heuristic was proposed by Hooker and Vinay [12]. It tries to avoid shortening many clauses by balancing the number of clauses that are shortened with the length to which they are shortened. Since it is bad to shorten clauses, they should be kept as long in size as possible until they are satisfied.

3.2 Simplification and Refutation Heuristics

Hooker and Vinay gave the following simplification hypothesis: *”Other things being equal, a branching rule performs better when it creates simpler subproblems”* [12]. It is motivated by an assumption that branching rules that produce simpler formulas produce smaller subtrees, which again are more likely to investigate smaller search spaces. This naturally gives us the strategy of choosing the subproblem with the smallest associated search tree. There are only two things that affect the size of a search tree, its depth and its branching factor. Since each level in the search tree for DPLL has the same branching factor of two, we only need to be concerned about the depth of the tree. The depth of a subtree is only decided by its number of variables, which again correlates with its number of clauses. This gives us the simplification strategy of choosing the subproblem with the smallest number of clauses.

In order to implement this strategy we can for each literal in our problem fix it, run propagation, and count the number of clauses. To avoid lookahead, we will use estimates of these numbers. We can use the number of unit clauses generated by propagation as our measuring stick on how many clauses are being satisfied (and thereby removed) and we get the maximum unit clause strategy of choosing the literal that spawns the most number of unit clauses through propagation. The measure $J(\neg l) - J(l)$ gives us an approximation of this.

Theorem 2. *Let $y(\mathcal{F})$ denote the size of the choice tree of $\mathcal{F}$. Given the formulae $\pi(\mathcal{F} \wedge l)$ and $\pi(\mathcal{F} \wedge l')$ where π includes unit propagation, we have* $y(\pi(\mathcal{F} \wedge l)) \leq y(\pi(\mathcal{F} \wedge l')) \Leftrightarrow J(\neg l) - J(l) \geq J(\neg l') - J(l')$.

For proof, see [12]. The larger the value of $J(\neg l) - J(l)$, the smaller is the subtree associated with branching on l. This gives us the following heuristic for sat problems:

The hvs heuristic [12]: Choose the literal with the highest value of $J(\neg l) - J(l)$, breaking ties randomly.

The hvs heuristic is designed to maximally remove literals and hence maximally reduce search trees sizes. It follows that the heuristic is a simplification heuristic. The history of the simplification strategy can be traced back to the *search rearrangement* heuristic [2]. Arguing that the search space could be reduced by creating subproblems with smaller search trees, the heuristic selects the subtree with the smallest number of remaining choices. Brelaz presented a heuristic for graph coloring that selects the most constrained node [3]. Haralick and Elliot described a CSP heuristic that became known as *fail first*: *"To succeed, try first where you are most likely to fail."* [11, page 263]. The "fail first" principle is basically a refutation strategy based on what we earlier have called the Refutation Hypothesis. We can easily design a first-order refutation heuristic by negating the first order satisfaction heuristic:

The first order refutation heuristic: Choose the literal with the highest value of $J(\neg l) - J(l)$, breaking ties randomly.

This heuristic is identical to the hvs heuristic. Fail first has been investigated in CSP [6; 7; 8], and we have seen published a simplification heuristic for sat based on search rearrangement [19], a simplification algorithm for CSP called *minimum width* ordering [10] and what is called a *fewest alternatives first* heuristic in planning [16; 21].

4 Experiments

4.1 Model

We will qualify the merit of a strategy by comparing it to the "don't care" strategy. In practice, this means that we take a heuristic that is a representative of the strategy and compare it to random selections over a representative set of sat problems. If it works better, it has merit. To do this, we need to define what we mean with "a representative of a strategy", "a representative set of sat problems", and how we compare heuristics.

What is a representative of a search strategy? Let us start with the satisfaction strategy. We define the solution frequency of a literal as the number of solutions the literal is participating in. Assuming independence we assume that the literal with the highest solution frequency is the literal with the highest probability for being part of a satisfiable extension of the partial solution.

Both the solution frequency and heuristics define an ordering over the literals. We will define a representative of the satisfaction strategy as a heuristic whose ordering is positively correlated with the frequency ordering. Similarly, we will define a good representative for the simplification strategy as a heuristic that has a positive correlation with subtree size. Furthermore, a good representative of a strategy shouldn't also be a representative for another strategy. If it were, we would not know which strategy its behavior could be contributed to. We will make the assumption that search space size and solution density are the only formula measures that have significant relevance to solution efficiency. Hence, a good representative for the satisfaction strategy is a heuristic that is positively correlated with solution frequency and not with sub-tree size, and a good representative for the simplification strategy is thus a heuristic that is positively correlated with search tree size and not with solution frequency.

Which problem classes do we investigate? Candidates for problem classes include real world problems modeled as sat problems, public benchmarks, and different models of randomly generated sat problems. We have chosen the latter. We will not argue that it is a representative class, but note that randomly generated sets makes it possible for us to control and vary parameters of interest and importance. We have chosen the *fixed clause length model* (also called the *fixed width model* or the *random k-sat model*). Given two positive integers k and n, assume the subset of sat consisting of all formulae with exactly n variables and k-length pure clauses. (A *pure* clause is a clause where each literal only occurs once.) The fixed clause length model aims at a uniform distribution of k-sat problems. Each clause is produced by sampling k times from a uniform distribution over the $2n$ available literals. This model is a popular distribution, because it is easier to generate hard instances belonging to this class. Instances from this distribution take exponential time on average for DPLL when finding all solutions [9]. The empirical performance of search algorithms over random k-sat problems has been investigated in several papers and interesting *phase transition* properties have been identified (see for instance [18]). Some of our measures are conditioned on the formulae actually having solutions. Due to the decreasing number of solvable formulae when density is increased, it is problematic to generate sufficiently large samples for problem densities above the phase transition. For these measures, we sometimes use *forced* formulae as source in addition to the normal formulae. Forced formulae are generated differently than the unforced, they are forced to have a solution. This is done by randomly determine a variable assignment B at the beginning of the generation process. During the clause generation process, only clauses which have B as a model are accepted for the formula F to be constructed. Forced distributions may have different characteristics than unforced and be easier to solve for some algorithms [13].

How do we compare heuristics? The ultimate meaning of a branching heuristic working better than another, is that it finds a solution faster. However, we have chosen *not* to use time as our measure as it is a problematic comparative measure (for supporting arguments, see for instance [12]). When comparing selection heuristics that have similar evaluation cost, a relevant alternative cost measure we will use is the number of nodes visited during the search .

4.2 The Simplification and Refutation Strategies

Where it matters, the hvs heuristic performs better than random, as can be seen from Figure 2. Hence, the heuristic has merit. We can see from Figure 3 that the hvs heuristic is negatively correlated with solution frequency. It follows that satisfaction can't be the reason for why hvs beats random, and simplification is the reason for its success. We can conclude that the simplification strategy has merit.

However, we can also see that though the heuristic beats random, it's not performing outstandingly. The gain over random selection is not substantial. In order to explain why, let us first discuss the refutation strategy.

In our opinion, the refutation strategy is not a sound principle for finding solutions, and we reject it. This position is supported by the fact that attempts at improving search by creating truer fail-first heuristics for CSP have rather resulted in poorer performance [20]. The simplification strategy is in contrast a sound strategy. This does however not mean that we will be able to find good simplification heuristics. The reason is that simplification in general is correlated with refutation. The hvs heuristic is exactly the opposite of the first order probability heuristic, which means that choosing according to this heuristic is choosing the subproblem with the smallest probability for containing a solution. The hvs heuristic is therefore both a representative for the simplification and refutation strategies, the latter keeping it from performing well.

The other way around, fail first is correlated with simplification. (As an example, when we designed a refutation heuristic we ended up with the hvs heuristic). Hence, it is not the refutation hypothesis that explains any successes the fail first principle has had, but rather the simplification hypothesis.

4.3 The Satisfaction Strategy

The satisfaction strategy will be supported if a satisfaction heuristic that is not positively correlated with simplification performs better than random. Figure 3 shows that the fop heuristic is positively correlated with solution frequency (though the strength of the correlation is density dependent) and we accept it as a representative of the satisfaction strategy.

Figure 2 shows that the fop heuristic generally performs better than random. In order to be able to support the satisfaction strategy, we need to make sure that the good performance of the fop heuristic is not because it correlates positively with simplification. We can not show that fop does not correlate positively with simplification; indeed, it does. (This is because satisfying many clauses usually also means fixing many literals, and hence shortening the search tree.) However, if we assume that the hvs is as good simplification as we can get, then there is obviously something else than simplification that has merit, since fop performs

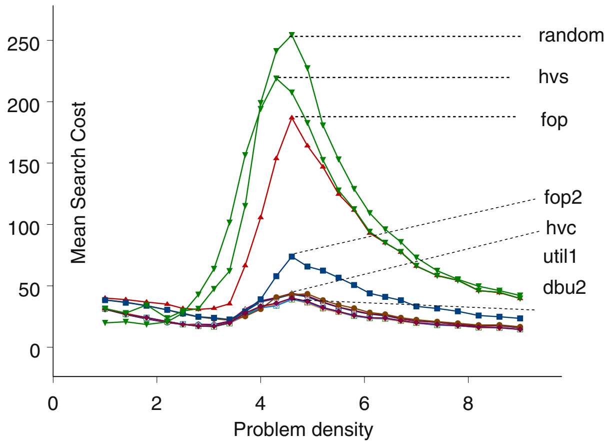


Fig. 2. Mean search cost (as mean number of literals fixed during search) as a function of density (the ratio of clauses to variables, m/n) of searching over 200 randomly generated 40 variable 3-sat formulae using DPLL with different heuristics.

better than hvs. We conclude that this is satisfaction, and henceforth that the satisfaction strategy has merit.

5 The Utility Strategy

Simplification (understood as fail first) has been claimed as better than satisfaction for csp [11], while it has been claimed as worse for sat [15].

It is argued that the reason why the jw heuristic works well is because it is a satisfaction heuristic [15; 12]. On the other hand, heuristics with more emphasis on solution probability (and stronger correlation with solution frequency) give poorer performance: Figure 3 shows that fop has a better correlation with solvability than jw while Figure 2 shows that fop performs poorer than jw. This seems to disqualify the satisfaction hypothesis.

In contrast, the simplification strategy was introduced in order to explain why jw's performs better than fop [12]. The argument is that jw is a simplification heuristic, and that simplification heuristics have better performance than satisfaction heuristics. But here as well, heuristics with more emphasis on simplification give poorer performance – Figure 2 shows that hvs performs poorer than jw. This seems to disqualify the simplification hypothesis.

Since fop is a purer satisfaction heuristics than jw, hvs a purer simplification heuristic, and jw performs better than either, an explanation has to be found in another strategy.

Let us create a heuristic by modifying the fop heuristic. In the analysis leading to the fop heuristic, it is assumed it is bad to shorten clauses, especially short

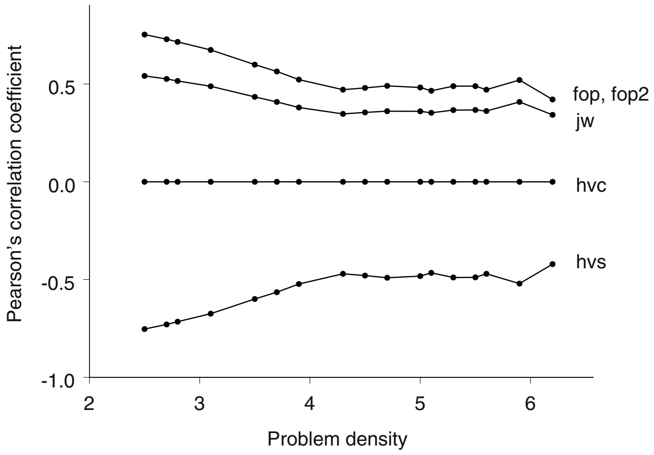


Fig. 3. The graphs show the mean Pearson's correlation coefficients between literal orderings according to solution frequency and orderings according to the fop, fop2, jw, hvc and hvs heuristics at the root nodes of 3-sat formulae. The correlations are shown as a function of density (m/n). Since we have a decreasing number of solvable formulae with increasing density, the data sets decrease and the means become less smooth.

clauses. On the other hand, a unit clause is automatically satisfied in the next round of propagation. Hence, a clause has a high probability of becoming satisfied both if it is kept long, and if it is shortened to a unit clause. (This is unless an inconsistency immediately appears between two unit clauses. However, detecting an inconsistency immediately in such a way is actually a good thing, since it is more efficient to detect errors early.) We can easily modify the fop heuristic to take this into account. In the following, let $J(\mathcal{F}_{\neg l}^{=2})$ be the subset of clauses that contains $\neg l$ and is of length 2, while $J(\mathcal{F}_{\neg l}^{>2})$ is the subset of clauses that contains $\neg l$ and is longer in size than 2. Figure 2 gives evidence that the resulting heuristic is a substantially better heuristic than fop:

The fop2 heuristic: *Choose the literal l with the highest value of $J(\mathcal{F}_l) + J(\mathcal{F}_{\neg l}^{=2}) - J(\mathcal{F}_{\neg l}^{>2})$, breaking ties randomly,*

This brings us closer to an understanding of why jw and fop2 work well. We suggest the following:

– *'Utility':* Hypothesis: Other things being equal, a branching rule performs better when it creates subproblems that have a good balance between satisfaction and simplification. Strategy: Select a subtree that balances satisfaction and simplification.

The efficiency of search must be understood as a utility function where the probability for reward is weighted against cost. In our context, these ingredients are in the shape of simplification and satisfaction. Pure satisfaction heuristics amount to utility functions where cost of error is neglected, while pure simplification heuristics amount to utility functions where solution probability is neglected.

We will now construct a monolithic utility heuristic based on the J-function. $J(l)$ counts contributions from satisfying clauses, which gives a measure of solvability. $J(\neg l)$ counts contributions from shortening clauses, which gives a measure of cost. Hence, a first attempt at a utility function, is simply:

THE FIRST UTILITY HEURISTIC: *(util1): Choose the literal with the highest value of* $\cos(\alpha\pi) \cdot J(l) + \sin(\alpha\pi) \cdot J(\neg l)$, *breaking ties randomly.*

The parameter α is used to assign different weights to the two terms. Figure 4 shows a polar log of the cost associated with varying α for 200 randomly generated 3-sat 40 variable formulae of density 4.3. Going around the circle takes you through different levels of satisfaction and simplification, and we find simplification in the second quadrant and satisfaction in the fourth. The performance is best in the first quadrant, and we can conclude that good heuristics need to both satisfy and simplify. Figure 4 b) shows a detailed plot of the first quadrant, giving evidence that the optimal value of α is 0.25. Interestingly, this gives us the following selection function, proposed in [12]. The heuristic behaves well, as can be seen in Figure 2.

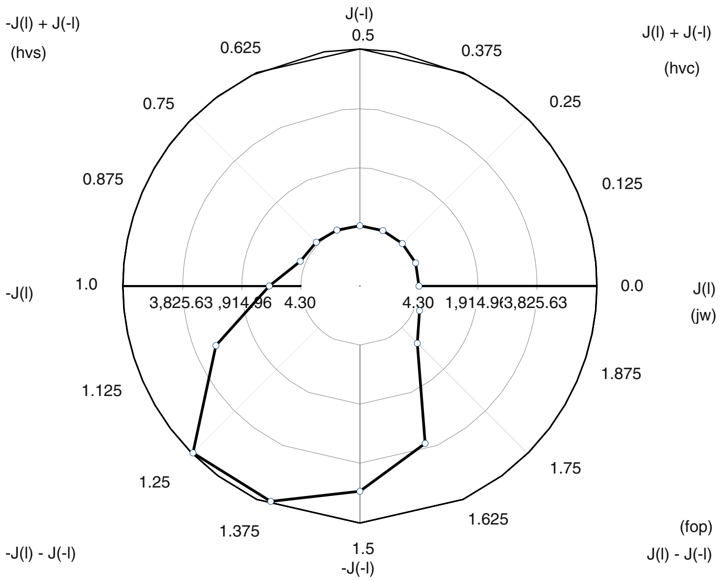
THE HOOKER-VINAY COMBINED HEURISTIC *(hvc) [12]: Choose the literal with the highest value of* $J(\neg l) + J(l)$, *breaking ties randomly.*

DPLL search trees are binary subtrees of the choice tree, where the two children of each node are a literal and its negation. If $\mathcal{A} \wedge l$ is false, then $\mathcal{A} \wedge \neg l$ must be true in order for $\mathcal{A}$ to be true, hence if a chosen sub-formula $\mathcal{F} \wedge l$ turns out not to be solvable, DPLL branches to $\mathcal{F} \wedge \neg l$. When choosing a literal we are choosing which literal to branch to in case it fails. Instead of having a choice between k variables and their negations (*i.e.*, $2k$ literals), we have a choice between k pairs of variables and their negations. Hence, we need a utility heuristic that order over pairs $(x, \neg x)$ instead of over literals.

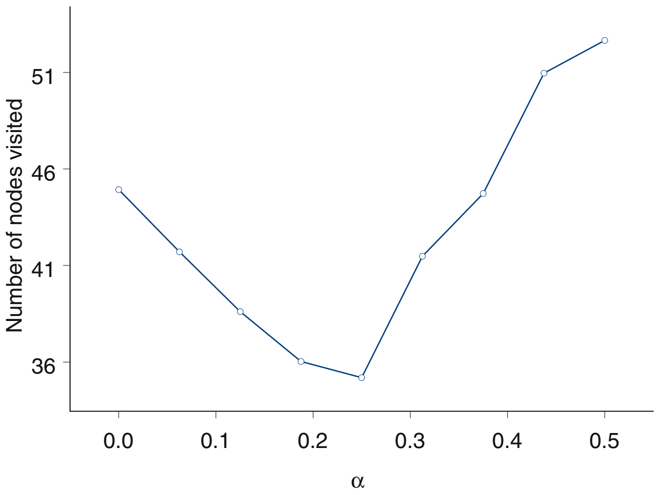
Theorem 3. *Given the formulae* $\mathcal{F}$ *and* $\mathcal{F}'$. *Let* $U(l) = \cos(\alpha\pi) \cdot P(l, \neg l) + \sin(\alpha\pi) \cdot y(l, \neg l)$, *where* $P(l, l')$ *is the probability that either* $\mathcal{F} \wedge l$ *or* $\mathcal{F} \wedge l'$ *have models, and* $y(l, \neg l)$ *is the combined size of the two subtrees* $y(\mathcal{F} \wedge l)$ *and* $y(\mathcal{F} \wedge l')$. *Then:*

$$U(l) \leq U(l') \Leftrightarrow J(l) \ + \ J(\neg l) \leq J(l') \ + \ J(\neg l'), \tag{2}$$

and



a) Polar cost plot. The axes show the mean search cost on a logarithmic scale, and the angles represent $\cos(\alpha \cdot \pi)$, *i.e.*, measured in radians. The numbers along the circumference are the values of α.



b) Zooming in at the first quadrant. Linear axes, α between 0.0 and 0.5.

Fig. 4. Mean search cost plot of *util1,* $\cos(\alpha \cdot \pi) \cdot J(l) + \sin(\alpha \cdot \pi) \cdot J(\neg l)$, at density 4.3 as a function of α. Each datapoint carries 200 40 variable 3-sat formulae.

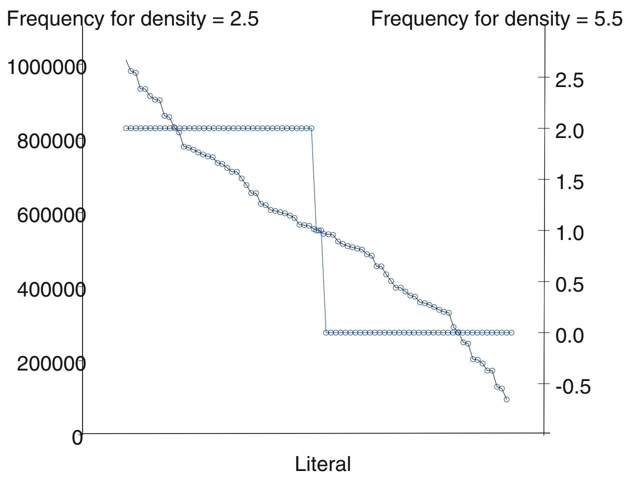


Fig. 5. Two single 40-variable, 100 clauses 3-sat instances are here investigated, one with density 1.0, another with density 2.2. The literals of the two instances are sorted along the x-axis according to decreasing solution frequency count. The y-axes show solution frequency for the literals. The y-axis to the left shows results for a single formula at density 2.5, the right y-axis shows the results for another single formula with density 5.5. We see that with increasing density, heuristics estimating frequency count (and hence solvability) become less discriminatory.

$$\begin{gathered} U(l) \le U(l') \Leftrightarrow \\ \cos(\alpha\pi) \cdot \mathtt{max}\{J(l), J(\neg l)\} + \sin(\alpha\pi) \cdot (J(l) + J(\neg l))\} \le \\ \cos(\alpha\pi) \cdot \mathtt{max}\{J(l'), J(\neg l')\} + \sin(\alpha\pi) \cdot (J(l') + J(\neg l'))\}. \end{gathered} \tag{3}$$

For proof, see [14]. $U(l)$ is the utility of a literal l as a chosen balance between the probability that the formula $\mathcal{F} \wedge l$ has a model and the size of the search tree associated with the formula. The cost if a selection fails is the size of the two subtrees. The balance of the subtrees has an impact on the combined size. Replacing two subtrees of size 2^p and 2^{p-2} with two of size 2^{p-1} gives a total reduction of 2^p nodes. The two equations in the Theorem lead us to the following two heuristics:

The first double branch utility heuristic: *(dbu1): Choose the literal with the highest value of* $\cos(\alpha\pi) \cdot J(l) \;+\; \sin(\alpha\pi) \cdot J(\neg l)$, *breaking ties randomly.*

The second double branch utility heuristic: *(dbu2): Choose the literal with the highest value of* $\cos(\alpha\pi) \cdot \mathtt{max}\{J(l), J(\neg l)\} \;+\; \sin(\alpha\pi) \cdot (J(l) + J(\neg l))$, *breaking ties randomly.*

We see that *dbu1* is equivalent to *util1*, which we arrived at through another argument, and seen that with optimal parameters it becomes the hvc heuristic, while Hooker and Vinay arrived at hvc through the double branch argument [12]. We

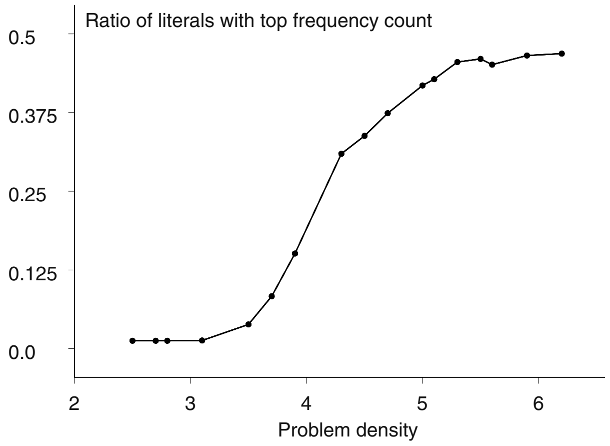


Fig. 6. Mean number of literals with the highest solution frequency at the root nodes of 40 variable 3-sat formulae as a function of density. These measure are only possible over solvable formulae. The solution frequency counts were generated by solving each formulae to completeness.

have seen that util1 performs well, and dbu2 performs as well or insignificantly better.

There is room for a possible improvement in double branch heuristics, since we sort according to tuples, but still need to choose which literal of the chosen tuple to branch to. An obvious method is to make this selection using the satisfaction strategy, for instance selecting between the two using fop:

The chained Hooker-Vinay combined heuristic (chvc) [12]: *Choose the literal l with the highest value of $J(l) + J(\neg l)$, then select between the two literals by choosing the one with the highest value of $J(l) - J(\neg l)$.*

The effect is however negligible in our experiments.

6 Conclusion

The satisfiability problem is central in complexity theory, artificial intelligence, mathematical logic and combinatorial optimization. This paper has discussed the k-sat subclass, where all clauses have equal length, and is an investigation into heuristics for the satisfiability problem and their relationship to time and hardness. An important algorithm for solving the sat problem is the DPLL algorithm. It can be seen as a constructive search algorithm that sequentially assigns truth values to boolean variables and backtracks when inconsistencies are encountered. Unit resolution is used as a propagation method after each time an assignment is made.

Search algorithms need, for any but the simplest problems, heuristics for guiding the selections made at each step in the search. The heuristics implement some basic strategy. We have identified a number of basic search strategies, and have decided that a strategy has merit if a representative heuristic of the strategy on average beats random selection. We chose the *fop* heuristic as a representative heuristic for the satisfaction strategy, and the *hvs* heuristic as a representative heuristic for the simplification strategy [12]. The satisfaction strategy is the strategy of searching where solutions are believed to most likely be. The simplification strategy is to search where the search spaces are believed to be smallest. The performances of the fop and the hvs heuristics indicate that the satisfaction and the simplification strategies both have merit for the k-sat problem. However, their merits are rather insignificant. A reason for this may be the special nature of the DPLL search trees, where you branch to the negation of the chosen literal when backtracking. This is especially troubling for fop and hvs, where an ordering, $l_1, \ldots, l_j$ implies that $l_i = \neg l_{n-i+1}$, i.e. when we backtrack we will branch into the worst possible subtree according to our ordering.

Good heuristics are hard to design. The analytical approach we have used in this paper comes with its own set of problems. First of all, the analytical heuristics necessarily make use of strong independence assumptions, in effect neglecting influence between literals. We see in Figure 3 that the correlation between solution frequency and fop decreases with increasing density. As the impact from other clauses increases with increased density, the independence assumption becomes unrealistic. It follows that the more constrained a problem is, the less useful are the analytical heuristics. As another example, the discriminatory power of analytical heuristics generally also diminishes with increased density. This can be seen from Figure 5, where a low and a high density 3-sat 40-variable formula are investigated. We se a steady decreasing frequency curve for the low density formula. For the high-density formula, however, we see that the set of literals actually are divided into two classes, those that are in a solution, and those that aren't. The *fop* heuristic would generally be able to choose a literal that is in a solution, but there is no discrimination between these literals. This point is supported in Figure 6, which shows that the mean number of literals with the highest frequency approaches 50% with increasing density.

We ended the paper by introducing the utility strategy, which advocates balancing the use of satisfaction and simplification. The utility strategy was found to be superior to the other strategies for the k-sat problem. Several utility heuristics have been designed. Furthermore, successful existing heuristics have been investigated, and their performance can now be explained through the notion of utility.

References

1. Bengt Aspvall, Michael F. Pass, and Robert Endre Tarjan. A linear time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, March 1979.

2. J. R. Bitner and E. M. Reingold. Backtrack programming techniques. *Communications of the ACM*, 18(11):651–656, November 1975.
3. D. Brelaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, April 1979.
4. Martin Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5:394–397, 1962.
5. Martin Davis and Hillary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960.
6. R. Dechter and J. Pearl. Generalized best-first search strategies and the optimality of A*. *Journal of the ACM*, 32(3):505–536, July 1985.
7. Rina Dechter and Judea Pearl. Network-based heuristics for constraint-satisfaction problems. *Artificial Intelligence*, 34(1):1–38, January 1988.
8. Rina Dechter and Judea Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38(3):353–366, April 1989.
9. John Franco and M. Paull. Probabilistic analysis of the Davis Putnam procedure for solving the satisfiability problem. *Discrete Applied Mathematics*, 5:77–87, 1983.
10. Eugene C. Freuder. A sufficient condition for backtrack-free search. *Journal of the ACM*, 29(1):24–32, January 1982.
11. R. Haralick and G. Elliot. Increasing tree search efficiency for constraint-satisfaction problems. *Artificial Intelligence Journal*, 14(3):263–313, 1980.
12. John N. Hooker and V. Vinay. Branching rules for satisfiability. *Journal of Automated Reasoning*, 15:359–383, 1995.
13. S. Hölldobler, H. Hoos, A. Strohmaier, and A. Weiß. The gsat/sa-familiy - relating greedy satisifability testing to simulated annealing, 1994.
14. Morten Irgens. *The Satisfiability Problem.* PhD thesis, Simon Fraser University, 2001. ISBN 3-89601-215-0.
15. Robert G. Jeroslow and Jinchang Wang. Solving propositional satisfiability problems. *Annals of Mathematics and Artificial Intelligence*, 1:167–187, 1990.
16. David Joslin and Martha Pollack. Least cost-flow repair: A plan refinement strategy for partial order planning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 1004–1009, Seattle, Washington, 1994. American Association for Artificial Intelligence.
17. D. W. Loveland. *Automated Theorem Proving: A Logical Basis.* North-Holland Publishing Co., Amsterdam, 1978.
18. David Mitchell, Bart Selman, and Hector Levesque. Hard and easy distributions of sat problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 440–446, Menlo Park, California, July 1992. The American Association for Artificial Intelligence.
19. Paul Walton Purdom. Search rearrangement backtracking and polynomial average time. *Artificial Intelligence Journal*, 21:117–133, 1983.
20. Barbara Smith and Stuart A. Grant. Trying harder to fail first. In Henri Prade, editor, *Proceedings of the European Confernece on Artificial Intelligence*, pages 249–253. The European Coordinating Committee on Artificial Intelligence, John Wiley & Sons, Inc., 1998.
21. Reiko Tsuneto, Dana Nau, and James Hendler. Plan-refinement strategies and search-space size. In *Proceedings of the fourth European conference on planning (ECP-97)*, 1997.

The Structural Model Interpretation of the NESS Test

Richard A. Baldwin and Eric Neufeld

Department of Computer Science, University of Saskatchewan,
Saskatoon, Canada SK S7H 3A8
{rab983,eric}@cs.usask.ca

Abstract. Within the law, the traditional test for attributing causal responsibility is the counterfactual "but-for" test, which asks whether the injury complained of would have occurred but for the defendant's wrongful act. This definition generally conforms to common intuitions regarding causation, but gives non-intuitive results in situations of overdetermination with two or more potential causes present. To handle such situations, Wright defined the NESS Test of causal contribution, described as a formalization of the concept underlying common intuitions of causal attribution. Halpern and Pearl provide a definition of actual causality in the mathematical language of structural models that yields counterintuitive results in certain scenarios. We present a new definition that appears to correct those problems and explain its greater conformity with the intuitions underlying the NESS test.

1 Introduction

According to Ashley (1990, p. 2), the legal domain is of interest to AI because it is between those formal domains so amenable to knowledge representation and those commonsense domains whose representation remains so elusive. Here we discuss *actual causation* (also called *causation in fact*) in the law from both perspectives. In common law jurisdictions, actual causation must be established between the defendant's wrongful conduct and the injury suffered for liability to be attached. The generally accepted test for determination of actual causation in the law is a counterfactual test, the *but-for* test. If the specified injury would not have occurred 'but for' the defendant's wrongful conduct, then actual causation is established. The test is considered straightforward enough to be applied by juries. However, the test is not comprehensive. It is known to fail when the scenario describing the injury includes other potential causes that would have caused the specified injury in the absence of the defendant's wrongful conduct. This is known as *overdetermined* causation.

Wright (1975, pp. 1775-76) divides cases of overdetermined causation into preemptive and duplicative causation cases. In preemptive causation, the effect of other potential causes is preempted by the effect of the defendant's wrongful act. For example, the defendant stabs and kills the victim before a fatal dose of poison previously administered by a third party can take effect. In duplicative causation, the effect of the defendant's act combines with, or duplicates, the effect of other potential causes where the latter were alone sufficient to bring about the injury. For example, the defendant and another party start separate fires that combine to burn down the victim's house where each fire was independently sufficient to do so. Since in these cases it is not true that 'but for' the defendant's wrongful act the specified harm would not have occurred, according to the but-for test, in neither scenario is the defendant's conduct an actual cause of the injury. Such a result is contrary to intuitions about responsibility and, by implication, about causality.

To cope with overdetermination, Wright (1985) proposes a comprehensive test for actual causation, the NESS (Necessary Element of a Sufficient Set) test. He adopts the view that there is an intelligible, determinate concept of actual causation underlying and explaining common intuitions and judgments about causality and that this concept explains the "intuitively plausible factual causal determinations" of judges and juries when "not confined by incorrect tests or formulas." Wright (1985, p. 1902) contends that not only does the NESS test capture the common-sense concept underlying these common intuitions and judgments, the NESS test defines that concept.

Pearl (2000, pp. 313-15) claims that while the intuitions underlying the NESS test are correct the test itself is inadequate to capture these intuitions because it relies on the traditional logical language of necessity and sufficiency, which cannot capture causal concepts. Pearl (Pearl, 1995; Galles and Pearl, 1997; Galles and Pearl, 1998; Pearl, 2000) proposes a mathematical language of structural causal models (structural language) for formalizing counterfactual and causal concepts. Pearl (1998; 2000, Chap. 10) first applies this structural language to define actual causation using a complex construction called a causal beam. (Halpern and Pearl, 2000) develops a "more transparent" definition (Halpern-Pearl definition), but still using structural models.

In (Baldwin and Neufeld, 2003) we suggested that the Halpern-Pearl definition essentially formalizes Wright's NESS test. However, a result of Hopkins and Pearl (2003) shows that this is not the case. In the sequel we discuss the implications of the Hopkins and Pearl result for the relationship between the Halpern-Pearl definition and the NESS test and, in response, we present an alternative structural language definition of actual causation which we believe does capture the essential meaning of the NESS test. We illustrate this through re-analysis of the examples in (Baldwin and Neufeld, 2003) and in the process lend validity to Wright's controversial NESS analysis of a complex class of causal scenarios known as double omission cases.

2 The NESS Test

Wright (1985, 1988, 2001) describes the NESS test as a refinement and development of the concept of a causally relevant condition as developed by Hart and Honore (1985). As Wright describes their analysis, this concept of singular (actual) causation depends on the core concept of general causality we all employ that conforms to and is explained by a regularity account of causation attributed to Hume as modified by Mill. To Hume is attributed the idea that causal attributions exhibit a belief that a succession of events fully instantiates some causal laws or generalizations (incompletely described causal laws). A causal law is described as an if-then statement whose antecedent lists minimal sufficient sets of conditions for (necessary for the sufficiency of) the consequent (the effect). Mill's contribution to the analysis is that there may be more than one set of sufficient conditions for an effect in general and in particular situations (the "plurality of causes" doctrine). Stated in full, the NESS test requires that a particular condition was a cause of (condition contributing to) a specific consequence if and only if it was a necessary element of a set of antecedent actual conditions that was sufficient for the occurrence of the consequence.

In circumstances where only one actual or potential set of conditions is sufficient for the result, the NESS test reduces to the but-for test (Wright, 1985). To illustrate that the NESS test matches common intuitions where the but-for test fails, Wright considers three variations of a two-fire scenario: fires X and Y are independently sufficient to destroy house H if they reach it and they are the only potential causes of house H's destruction so that if neither reach the house it will not be destroyed. In the first situation, X reaches and destroys H and Y would not have reached H even if X were absent. The common intuition here is that X was a cause of the destruction of H but not Y. In this case there is a single actually sufficient set of conditions and no other even potentially sufficient set of conditions. (This assumes that actually sufficient sets of conditions are minimal.) X was a necessary element (necessary for the sufficiency) of that single, actually sufficient set, a NESS condition. It was also a but-for condition.

In the second situation, X and Y reach H simultaneously and combine to destroy it. Here Wright claims that the common intuition is that both X and Y were causes of the destruction of the house. There are two overlapping sets of actually sufficient conditions. X is necessary for the sufficiency of the set including itself but not Y and Y is necessary for the sufficiency of the set including itself but not X. Neither X nor Y is a but-for cause of the destruction of H but each is a duplicative NESS cause.

In the final situation, X reaches and destroys H before Y can arrive and, if X had been absent, Y would have destroyed H. Here the common intuition is unquestionably that X caused the destruction of H and Y did not. Fire Y is not a NESS condition for the destruction of H since any actually sufficient set of conditions, given the assumptions of the scenario, must include X, and Y is not necessary for the sufficiency of any set of conditions that includes X. Fire X, on the other hand, is necessary for the sufficiency of the actually sufficient set of which it is a member. Because the set containing Y but not X would have been sufficient in the absence of X, X is not a but-for cause of the destruction of H. X was a preemptive NESS cause because it preempted the actual sufficiency of the potentially sufficient set including Y.

3 The Structure Equation Model

Following (Halpern and Pearl, 2001; Pearl, 2000) a *signature* $\mathcal{S}$ is a 3-tuple ($\mathcal{U}$, $\mathcal{V}$, $\mathcal{R}$), where $\mathcal{U}$ is a finite set of exogenous variables, $\mathcal{V}$ is a set of endogenous variables, and $\mathcal{R}$ is a relation associating with each variable $Y \in \mathcal{U} \cup \mathcal{V}$ a nonempty set $\mathcal{R}(Y)$ of possible values for Y (the *range* of Y).

A *causal model* over a signature $\mathcal{S}$ is a 2-tuple $M = (\mathcal{S}, \mathcal{F})$, where $\mathcal{F}$ is a relation associating each $X \in \mathcal{V}$ with a function F_X that describes the outcome of X given the values of other variables in the model. These functions are the structural equations and describe the process by which the dependent variable receives its value; they correspond to causal mechanisms (law-like regularities) in the domain being modeled. The value of endogenous variables are determined by the values of other endogenous variables and exogenous variables. The value of exogenous variables are determined outside the model and are given. A particular setting $\vec{u}$ of variables in $\mathcal{U}$ is a *context* for the model M and a model with a given context is a *causal world*.

We consider *recursive* causal models, where for any two variables X and Y either F_X is independent of the value of Y (i.e., $F_X(\ldots, y, \ldots) = F_X(\ldots, y', \ldots)$ for all $y, y' \in \mathcal{R}(Y)$) or F_Y is independent of the value of X. Recursive causal models have a unique solution, a unique set of variable values simultaneously satisfying all model equations. If PA_X is the minimal set of variables in $\mathcal{V} - X$ and U_X the minimal set of variables in $\mathcal{U}$ that together suffice to represent F_X, a recursive causal model gives rise to a *causal diagram*, a directed acyclic graph (DAG) where each node corresponds to a variable in $\mathcal{V} \cup \mathcal{U}$ and the directed edges point from members of $PA_X \cup U_X$ to X and are *direct causes* of X. (PA_X connotes the *parents* of X, conventionally restricted to endogenous variables.) The edges in a causal diagram represent the non-parameterized (or arbitrary) form of the function for a variable, $X = F_X(U_X, PA_X)$.

An external *intervention* (or *surgery*) setting $X = x$, where $X \in \mathcal{V}$, is denoted $X \leftarrow x$ and amounts to pruning the equation for X from the model and substituting $X = x$ in the remaining equations. An intervention that forces the values of a subset of $\mathcal{V}$ prunes a subset of equations from the model, one for each variable in the set, and substitutes the corresponding forced values in the remaining equations. (A set X of variables in $\mathcal{V}$ is sometimes written $\vec{X}$ and a setting of those vectors is written $\vec{X} \leftarrow \vec{x}$.) Interventions represent non-modeled contingencies perturbing causal mechanisms. The result of an intervention $\vec{X} \leftarrow \vec{x}$ is a new causal model (a *submodel*), denoted $M_{\vec{X} \leftarrow \vec{x}}$, over the signature $\mathcal{S}_{\vec{X}} = (\mathcal{U}, \mathcal{V} - \vec{X}, \mathcal{R}|_{\mathcal{V} - \vec{X}})$. In the corresponding causal diagram, it amounts to removing the edges from $PA_X \cup U_X$ to X. A submodel represents a counterfactual world.

For a given signature $\mathcal{S} = (\mathcal{U}, \mathcal{V}, \mathcal{R})$, a *primitive event* is a formula of the form $X = x$, where $X \in \mathcal{V}$ and $x \in \mathcal{R}(X)$. A *basic causal formula* is of the form $[Y_1 \leftarrow y_1, \ldots, Y_k \leftarrow y_k]\varphi$, where φ is a Boolean combination of primitive events, $Y_1 \ldots Y_k$ are distinct variables in $\mathcal{V}$, and $y_i \in \mathcal{R}(Y_i)$. Basic causal formulas are abbreviated as $[\vec{Y} \leftarrow \vec{y}]\varphi$ or just φ when $k = 0$. A *causal formula* is a Boolean combination of basic causal formulas.

A basic causal formula is true or false in a causal model given a context $\vec{u}$. Where ψ is a causal formula (or a Boolean combination of primitive events), $(M,\vec{u}) \models \psi$ means that ψ is true in the causal model M in the context $\vec{u}$. $(M,\vec{u}) \models [\vec{Y} \leftarrow \vec{y}](X = x)$ means that X has value x in the unique solution to the equations in the submodel $M_{\vec{Y} \leftarrow \vec{y}}$ in context $\vec{u}$. In other words, in the world in which $\mathcal{U} = \vec{u}$, the model predicts that if $\vec{Y}$ had been $\vec{y}$ then X would have been x; that is, in the counterfactual world $M_{\vec{Y} \leftarrow \vec{y}}$, resulting from the intervention $\vec{Y} \leftarrow \vec{y}$, X has the value x. Causes are conjunctions of primitive events of the form written $\vec{X} = \vec{x}$.

Definition (*Actual Cause; Halpern-Pearl*): $\vec{X} = \vec{x}$ is an *actual cause* of φ in a model M in the context $\vec{u}$ (i.e., in $(M,\vec{u})$) if the following conditions hold:

HC1. $(M,\vec{u}) \models (\vec{X} = \vec{x}) \wedge \varphi$.

HC2. There exists a partition $(\vec{Z},\vec{W})$ of $\mathcal{V}$ with $\vec{X} \subseteq \vec{Z}$ and some setting $(\vec{x}',\vec{w}')$ of the variables in $(\vec{X},\vec{W})$ such that, where $(M,\vec{u}) \models Z = z^*$ for each $Z \in \vec{Z}$ (i.e., the actual value of z in context $\vec{u}$

(a) $(M,\vec{u}) \models [\vec{X} \leftarrow \vec{x}', \vec{W} \leftarrow \vec{w}']\neg\varphi$, and

(b) $(M,\vec{u}) \models [\vec{X} \leftarrow \vec{x}, \vec{W} \leftarrow \vec{w}', \vec{Z}' \leftarrow \vec{z}^*]\varphi$ for every subset $\vec{Z}'$ of $\vec{Z}$.

HC3. $\vec{X}$ is minimal; no subset of $\vec{X}$ satisfies conditions HC1 and HC2.

Condition HC2 exemplifies what has been called the "counterfactual strategy" for defining actual causation—"event C causes event E *iff* for some *appropriate* G, E is counterfactually dependent on C when we hold G fixed" (Hopkins and Pearl, 2003). The basic idea is that an active causal process (a set Z satisfying condition HC2) be shielded from spurious (preempted) or redundant (duplicative) causal processes before testing whether the effect is counterfactually dependent on the putative cause. However, contrary to the claim in (Baldwin and Neufeld, 2003) that the Halpern-Pearl definition essentially formalizes Wright's NESS test, it turns out that the choice of "appropriate G" under condition HC2 is too permissive to represent the NESS test in the language of structural models.

4 The Halpern-Pearl Definition and NESS

Hopkins and Pearl (2003) show that even locally, between a variable (the effect) and its parents (direct causes), the Halpern-Pearl definition does not require that an active causal process (a set Z) be actually sufficient; counterfactual dependence satisfying condition HC2(b) may depend on *non-actual* conditions.

To see this, consider a causal model M with context $\mathcal{U} = \vec{u}$. For simplicity, assume that all $U \in \mathcal{U}$ are trivial in the structural equation F_X for X (i.e., U_X is empty) so that $F_X : Dom(PA_X) \rightarrow Dom(X)$. Now consider an assignment $\overrightarrow{PA_X} \leftarrow \overrightarrow{pa_X}$ (here, the vector notation represents an ordered assignment of values $\overrightarrow{pa_X}$ to variables $\overrightarrow{PA_X}$) such that $X = x$. If $\overrightarrow{PA_X} = \{V_1,\ldots,V_n\}$ and $\overrightarrow{pa_X} = \{v_1,\ldots,v_n\}$, the logical sentence consisting of the conjunction of literals $V_i = v_i$ (i.e., $V_1 = v_1 \wedge \ldots \wedge V_n = v_n$) implies $X = x$. If such a sentence is formed for each assignment $\overrightarrow{PA_X} \leftarrow \overrightarrow{pa_X}$ such that $X = x$ and a new sentence, denoted $\Delta(X = x)$, is formed as a disjunction of all such sentences (so that the resulting logical sentence is in disjunctive normal form), then $X = x$ *iff* $\Delta(X = x)$. To illustrate, Hopkins and Pearl give this example.

Example (firing squad) A firing squad consists of shooters B and C. Shooter C loads and shoots his own gun while shooter B is lazy and insists that A load his gun for him. The prisoner D will die (D = 1) if, and only if, either A loads B's gun (A = 1) and B shoots (B = 1) or C loads his gun and shoots (C = 1); that is, $D = (A \wedge B) \vee C$.

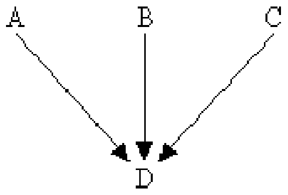


Fig. 1. Firing Squad

For this example, if $(M,\vec{u}) \models (D = 1)$ then

$$\Delta(D = 1) = (A = 1 \wedge B = 1 \wedge C = 1) \vee (A = 1 \wedge B = 1 \wedge C = 0) \vee (A = 0 \wedge B = 1 \wedge C = 1)$$
$$\vee (A = 1 \wedge B = 0 \wedge C = 1) \vee (A = 0 \wedge B = 0 \wedge C = 1) .$$

A term (conjunction of literals) that entails a sentence S is an *implicant* of S; an implicant that does not entail any other implicant is a *prime implicant*. The *prime implicant form* of a sentence is a disjunction of all its prime implicants and is unique. The prime implicant form of $\Delta(D = 1)$ is $\Delta(D = 1) = (A = 1 \wedge B = 1) \vee (C = 1)$.

With these preliminaries, Hopkins and Pearl (2003) prove the following theorem:

Theorem (prime implicant) In a causal model M with context $\mathcal{U} = \vec{u}$, let $X, Y \in \mathcal{V}$ with $X \in PA_Y$. If $(M,\vec{u}) \models (X = x \wedge Y = y)$ and the literal $X = x$ occurs in any prime implicant of $\Delta(Y = y)$ then the Halpern-Pearl definition of actual causation will classify $X = x$ as an actual cause of $Y = y$.

Note that the prime implicant theorem does *not* require that any other literals (if they exist) in any of the prime implicants of $\Delta(Y = y)$ to which $X = x$ belongs be satisfied (true) in $(M,\vec{u})$. For example, assume the context $\vec{u}$ in the firing squad example is such that C shoots and A loads B's gun, but B does not shoot. Since

$(M,\vec{u}) \models (A=1 \wedge D=1)$ and $A=1$ occurs in the prime implicant $(A=1 \wedge B=1)$ for $\Delta(D=1)$, according to the prime implicant theorem, the Halpern-Pearl theorem should (counter-intuitively) classify *A*'s loading of *B*'s gun as a cause of *D*'s death though *B* does not shoot. Indeed, taking $\vec{Z}=(A,D)$ and $\vec{W}=(B,C)$ and setting $\vec{W}=\vec{w}$ = (1,0)) satisfies conditions HC2(a) and (b) of the definition.

Hopkins and Pearl (2003) point out the similarity of the prime implicant form of a sentence with Mackie's INUS account of singular causation, according to which *C* is a cause of *E* iff *C* is an insufficient but non-redundant part of an unnecessary but sufficient condition for *E* (Mackie, 1974, pp. 61-62): "For instance, *A* loading *B*'s gun is a necessary part of a sufficient condition to ensure the prisoner's death. In terms of the prime implicant logical form, sufficient conditions map to implicants. For instance, $A=1 \wedge B=1$ is a sufficient condition for $D=1$. Furthermore, since $A=1 \wedge B=1$ is a *prime* implicate (hence no subset of its conjuncts is an implicate), we observe that both $A=1$ and $B=1$ are necessary parts of this sufficient condition. Hence any atomic expression that appears in a prime implicate satisfies the INUS condition."

Accepting the mapping of sufficient conditions to implicants, then Mackie's INUS test (Kim, 1993) requires that for $A=1$ to be a cause of $D=1$, not only must $A=1$ occur as an atomic proposition (or literal) in some prime implicant for $D=1$ (i.e., be an INUS condition for $D=1$) but also that every other atomic proposition in that implicant be satisfied. This part of Mackie's analysis is consistent with the NESS test (see Baldwin, 2003). It follows then that the Halpern-Pearl definition is less restrictive than both Mackie's INUS analysis and Wright's NESS test. As Hopkins and Pearl say, their prime implicant theorem exposes that the Halpern-Pearl definition is over permissive. It is at least too permissive to formally capture the meaning of the NESS test.

5 Preliminaries to the New Definition

We take as a consequence of Hopkin's and Pearl's prime implicant theorem that if a structural definition of actual causation employing the counterfactual strategy is to capture the meaning of the NESS test, the choice of which variables in the model may have their values fixed must be controlled by the relationship of belonging to the same minimal sufficient condition (set of conditions) for an effect. Pearl (2000) suggests that this information may already be encoded in the structural language. In discussing Mackie's (1974) scenario of the ill-fated desert traveler—where a traveler has two enemies, one who poisons (p = 1; variables are chosen to be consistent with Pearl's exposition) the traveler's water canteen and the other, unaware of the poisoning, shoots and empties the traveler's canteen (x = 1) as a result of which the traveler dies—Pearl (2000, p. 312) considers the structural equations $y = x \vee px'$ (x' is equivalent to $\neg x$) and the *logically* equivalent $y = x \vee p$ and states, "Here we see in vivid symbols the role played by structural information. Although it is true that

$x \vee x'p$ is logically equivalent to $x \vee p$, the two are not structurally equivalent; $x \vee p$ is completely symmetric relative to exchanging x and p, whereas $x \vee x'p$ tells us that, when x is true, p has no effect whatsoever—not only on y, but also on any of the intermediate conditions that could potentially affect y. It is this asymmetry that makes us proclaim x and not p to be the cause of death."

Hausman and Woodward (1999) shed more light on the relationship between the structural information that the causal relation between an independent variable and dependent variable in a structural equation depends (or does not depend) on one or more other independent variables in the equation (e.g., the difference between $x \vee p$ and $x \vee x'p$) and minimal sets of sufficient conditions. They point out that a structural equation may capture more than one causal mechanism; terms in additive relations in a single structural equation may represent distinct causal mechanisms. For example, x and $x'p$ represent distinct causal processes ending in the traveller's death. Hausman and Woodward (1999, p. 547) say that a system of structural equations with additive terms where each *term* in each equation represents a distinct causal mechanism ("and thus acts independently of the mechanisms registered by other terms") exhibits *coefficient invariance*, which is violated when the terms in the structural equations are not additive or when the causal relationship between two variables depends on the level of a third. Hausman and Woodward (1999, p. 547) then say, "If one thinks of individual causes of some effect Y as conjuncts in a minimal sufficient condition for Y (or the quantitative analogue thereof)—that is, as 'conjunctive causes'—then the relationship between an effect and its individual causes will not satisfy coefficient invariance"

Hausman and Woodward define coefficient invariance as a global property of systems of structural equations and as a means of identifying distinguishable (if not distinct in the sense of shared variables) mechanisms within a single structural equation—which we require—their explication of the concept is too strict. (Hausman and Woodward (1999, p. 547) say that coefficient invariance holds only when *every* variable in a structural equation belongs to a single additive term ruling out, for example, equations of the form $y = x \vee px'$.) For our purposes, it is enough that for a set of additive structural equations, expressed in sum of products form, each term in an equation represents a separate mechanism, a separate set of minimal sufficient conditions ("or the quantitative analogue thereof"), for the effect represented by the equation's dependent variable. We call this property *term modularity*. A causal model whose structural equations satisfy this property allows for the development of a criterion, ultimately derivative of Hausman and Woodward's concept of coefficient invariance, for determining what variables should be held fixed and what variables may be altered in testing for counterfactual dependence between an effect variable and one of its (putative) causal variables in the model. This, in turn, will allow for a new structural definition of actual causation that avoids the problem identified by Hopkins and Pearl (see Section 4) and that formalizes the NESS test in the context of a scenario modelled by a causal world in the structural language.

The "term modularity" criterion encodes the relationship, among the parents of a variable, of being components of distinct component causal mechanisms (or elements of minimal sufficient conditions for the variable). For distinct variables *X*, *Y*, and *Z*, where *X* and *Y* occur as independent variables in the structural equation (parents) for Z in a causal world $(M,\vec{u})$, X *is coefficient invariant to* Y *for term* T if $(M,\vec{u}) \models \neg(T=0)$ (i.e., the term is satisfied, or non-zero in quantitative contexts, in $(M,\vec{u})$), *X* occurs as a literal in *T*, and *Y* is not a variable in *T* (symbolically, $coin_{\vec{u}}^{T}(Z;X\,|\,Y)$; note that *X* is a literal of the form $X=x$ where $(M,\vec{u}) \models (X=x)$ while *Y* is a variable). When $coin_{\vec{u}}^{T}(Z;X\,|\,Y)$ the causal relation between *X* and *Z* does not depend on the value of *Y* in context $\vec{u}$. Locally, to avoid satisfying actually unsatisfied terms (minimal sufficient sets) as happens with the Halpern-Pearl definition, between a variable *X* and its parents ($Y_1,\ldots,Y_n$), in testing whether $Y_i = y_i$ is an actual cause of *X* = *x*, $Y_i = y_i$ should belong to a satisfied term *T* and only parent variables Y_j that Y_i is coefficient invariant to for *T* in the equation for *X* ($coin_{\vec{u}}^{T}(X;Y_i\,|\,Y_j)$) should be allowed to have their values altered.

Returning to Hopkins and Pearl's firing squad example (see Section 4), with structural equation $D=(A \wedge B) \vee C$, $(M,\vec{u}) \models (A \wedge B)$ and it is not the case that *A* is coefficient invariant to *B* for $T=(A \wedge B)$ in the equation for *D* ($\neg coin_{\vec{u}}^{(A\wedge B)}(D;A\,|\,B)$). Therefore, in the context such that *A* = *C* = 1 and *B* = 0, to test whether *D* = 1 is counterfactually dependent on *A* = 1, the value of *B* may not be altered. Since $(A \wedge B)$ is the only satisfied term in the equation for *D* in which *A* (i.e., $A=1$) occurs, contrary to the Halpern-Pearl approach, it is not possible to modify the model so that *D* is counterfactually dependent on *A* in a scenario where *B* = 0.

Globally in a causal model with context $\mathcal{U} = \vec{u}$, when distinct variables Y_1 and Y_2 occur as common parents of distinct variables X_1 and X_2 it can happen that there exists a term T_{X_1} in the equation for X_1 such that $coin_{\vec{u}}^{T_{X_1}}(X_1;Y_1\,|\,Y_2)$ but any satisfied term T_{X_2} in the equation for X_2 that includes Y_1 also includes Y_2 ($\neg coin_{\vec{u}}^{T_{X_2}}(X_2;Y_1\,|\,Y_2)$); that is, Y_1 is coefficient invariant to Y_2 for some term in the equation for X_1 but not in the equation for X_2. In that case, if Y_1 is part of the "active causal process" being tested, before allowing the value of Y_2 to be altered, it is necessary to interfere directly in the equation for X_2 by substituting a constant y_2 for Y_2 in the equation for X_2 where $Y_2 = y_2$ in the unaltered model (i.e., fix Y_2 at its actual value, $(M,\vec{u}) \models (Y_2 = y_2)$). This avoids the possibility of the counterfactual or original values of Y_1 interacting with non-actual values to satisfy non-actually satisfied minimal sufficient sets for some variable, the problem that plagues the Halpern-Pearl definition. This process must be repeated for all X_i where for all satisfied term T_{X_i} including $\neg coin_{\vec{u}}^{T_{X_i}}(X_i;Y_1 \mid Y_2)$. Only then should altering the value of Y_2 be allowed.

6 New Structural Definition of Actual Causation

A *causal route* $\vec{R} = \langle C, D_1, \ldots, D_n, E \rangle$ between two variables C and E in $\mathcal{V}$ is an ordered sequence of variables such that each variable in the sequence is in $\mathcal{V}$ and a parent of its successor in the sequence.

For a causal mode M with route $\vec{R} = \langle C, D_1, \ldots, D_n, E \rangle$ and a sequence of terms $\vec{T} = \langle T_{D_1}, \ldots, T_{D_n}, T_E \rangle$, where T_X is a satisfied term in the equation for X, the *sub-model relative to* $\vec{R}$ *and* $\vec{T}$ *in context* $\vec{u}$ (denoted $M^{\vec{T}}_{[\vec{R},\vec{u}]}$) is derived from $(M,\vec{u})$ as follows: for distinct $X,Y,W \in \mathcal{V}$ with $X \in \vec{R} - E$, $Y \notin \vec{R}$, and $W \neq C$, if $\neg coin_{\vec{u}}^{T_W}(W;X \mid Y)$ replace the function F_W for W (see Section 3) by the function that results when V is replaced with a constant v where $(M,\vec{u}) \models (V = v)$.

Definition (*actual cause*; *new version*) $C = c$ is an actual cause of $E = e$ in $(M,\vec{u})$ if the following conditions hold:

AC1. $(M,\vec{u}) \models (C = c \wedge E = e)$

AC2. There exists a route $\vec{R} = \langle C, D_1, \ldots, D_n, E \rangle$ in M, a sequence of satisfied terms $\vec{T} = \langle T_{D_1}, \ldots, T_{D_n}, T_E \rangle$, and a setting $\vec{w}$ for $\vec{W} = \mathcal{V} - \vec{R}$ and a setting $c' \neq c$ for C such that:

(a) $(M^{\vec{T}}_{[\vec{R},\vec{u}]}, \vec{u}) \models [C \leftarrow c', \vec{W} \leftarrow \vec{w}]\neg(E = e)$, and

(b) $(M^{\vec{T}}_{[\vec{R},\vec{u}]}, \vec{u}) \models [C \leftarrow c, \vec{W} \leftarrow \vec{w}](E = e)$.

Because there are no causal interaction effects between variables in $\vec{R}$ and $\vec{W}$ in $M^{\vec{T}}_{[\vec{R},\vec{u}]}$, by the construction of $M^{\vec{T}}_{[\vec{R},\vec{u}]}$ (variables in $\vec{R}$ are coefficient invariant to all variables in $\vec{W}$ by definition of $M^{\vec{T}}_{[\vec{R},\vec{u}]}$), the setting $\vec{W} \leftarrow \vec{w}$ cannot "contaminate" the test of counterfactual dependence in AC2 in the sense of satisfying a non-actually satisfied minimal sufficient set of conditions.

In practice, it rarely happens that a literal X occurs in more than one satisfied term in a structural equation; a non-quantitative equation having more than one satisfied term with distinct literals only occurs itself in cases of duplicative causation. To avoid the cumbersome and somewhat confusing terminology, subsequently, unless the context requires otherwise (as in the analysis of the pollution cases in Section 7.2), the choice of the sequence $\vec{T}$ will be left as implied by the analysis of the scenario and the superscript $\vec{T}$ left out of the notations $coin_{\vec{u}}^{T}(Z;X \mid Y)$ and $M_{[\vec{R},\vec{u}]}^{\vec{T}}$.

Suppose that C is an actual cause of E in $(M,\vec{u})$. Then there exists a route $\vec{R} = \left\langle C, D_1, \ldots, D_n, E \right\rangle$ and a sequence of satisfied terms $\vec{T} = \left\langle T_{D_1}, \ldots, T_{D_n}, T_E \right\rangle$ satisfying condition AC2 of the new definition of actual causation. The *active causal process relative to* $\vec{R}$ *and* $\vec{T}$ *in* $\vec{u}$ (denoted $ACP_{[\vec{R},\vec{u}]}^{\vec{T}}$) is the set $\vec{R} \cup \{X_i\}$ where $X_i \in \mathcal{V} - \vec{R}$, $Y \in \vec{R} - E$, $Z \in \vec{R} - A$, and $\neg coin_{\vec{u}}^{\vec{T}Z}(Z;Y \mid X_i)$. That is to say, $ACP_{[\vec{R},\vec{u}]}^{\vec{T}}$ is the subset of variables in $\mathcal{V}$ that have their valued fixed in forming $M_{[\vec{R},\vec{u}]}^{\vec{T}}$ that are parents of a variable in $\vec{R}$. (Again, the term specific terminology and the accompanying superscripts will be discarded where the context does not require them.)

7 Examples of NESS and the New Definition

7.1 Preemptive Causation

Wright (1985, p 1795) considers two scenarios: in the first, *D* shoots and kills *P* before *P* drinks tea fatally poisoned by *C* and, in the second, *D* shoots and instantly kills *P* after *P* drinks tea fatally poisoned by *C* but before the poison takes effect. In the first scenario, in Wright's (1985, p 1795) NESS analysis, *D*'s shot was necessary for the sufficiency of a set of actual antecedent conditions that did not include the poisoned tea. Conversely, *C*'s poisoning of the tea was not a necessary element of any sufficient set of actual antecedent conditions. A set that included the poisoned tea but not the shooting would be sufficient only if *P* actually drank the tea, but this was not an actual condition. The shooting preempted the potential causal effect of the poisoned tea.

In this scenario, the story of death by poisoning would have occurred (the intake of the poison through consumption of the tea will have occurred) but for *D* shooting *P*. This is reflected in the following causal model. The model has the following propositional variables: *DS* represents "*D* shoots," *PT* represents "*C* poisons the tea," *CP* represents "*P* consumes poison," and *PD* for "*P* dies." The structural equations are $CP = \neg DS \wedge PT$ and $PD = DS \vee CP$. The causal diagram corresponding to these equations is represented in Figure 2.

To show that $DS = 1$ is an actual cause of $PD = 1$, let $\vec{R} = \left\langle DS, PD \right\rangle$ for condition AC2. Since $coin_{\vec{u}}(PD;DS \mid CP)$, $M_{[\vec{R},\vec{u}]} = (M,\vec{u})$ and therefore $\vec{W} = (PT, CP)$.

Setting $\vec{W} = (0,0)$ then satisfies conditions AC2(a) and (b). Note that $ACP_{[\vec{R},\vec{u}]} = \vec{R}$ and the NESS set including $DS = 1$ for $PD = 1$ in $(M, \vec{u})$ is just $\{DS = 1\}$, as it is with Wright's analysis.

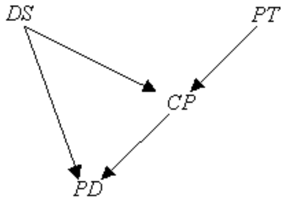


Fig. 2. Causal diagram for the poisoned tea scenario

Suppose that the context was such that *D* does not shoot ($\neg DS$) but P still poisons the tea. Then $CP = 1$ and $PD = 1$ and it is straightforward to show that $PT = 1$ is a cause of $PD = 1$ by letting $\vec{R} = \langle PT, CP, PD \rangle$ in condition AC2. Note, however, that since $\neg coin_{\vec{u}}(CP; PT \mid DS)$, $ACP_{[\vec{R},\vec{u}]} = \{PT, CP, DS, PD\}$ and the NESS set including $PT = 1$ for $PD = 1$ in $(M, \vec{u})$ is $\{PT = 1, CP = 1, DS = 0\}$: the absence of the preempting condition *DS* must be included.

For the second example, Wright's (1985, p 1795) NESS analysis of why *D*'s shooting was a cause of *P*'s death is the same as that for the first example; as to whether *C*'s poisoning of the tea was a cause: "Even if P actually had drunk the poisoned tea, C's poisoning of the tea still would not be a cause of P's death if the poison did not work instantaneously but the shot did. The poisoned tea would be a cause of P's death only if P drank the tea and was alive when the poison took effect. That is, a set of actual antecedent conditions sufficient to cause P's death must include poisoning of the tea, P's drinking the poisoned tea, and P's being alive when the poison takes effect. Although the first two conditions actually existed, the third did not. D's shooting P prevented it from occurring. Thus, there is no sufficient set of actual antecedent conditions that includes C's poisoning of the tea as a necessary element. Consequently, C's poisoning of the tea fails the NESS test. It did not contribute to P's death."

A causal model for this scenario differs from the previous one by the addition of a variable *PTE* for "the poison takes effect." The structural equation for *PD* becomes $PD = DS \vee PTE$ and the equation for *CP* becomes $CP = PT$. As with Wright's NESS analysis, the proof that $DS = 1$ is an actual cause of $PD = 1$ would be essentially the same as with the previous example.

7.2 Duplicative Causation Scenarios

Among the duplicative causation cases, of particular interest are a group of pollution cases where defendants were found liable though none of their individual acts (their "contributions" to the pollution) was sufficient, or necessary given the contributions of the other defendants, to produce the plaintiff's injuries (some adverse effect on the use of his property).[1] Wright (1985, p 1793) applies the NESS test to an idealized example in which, five units of pollution are necessary and sufficient for the plaintiff's injury and seven defendants discharge one unit each. The NESS test requires only that a defendant's discharge be necessary for the sufficiency of *a* set of actual antecedent conditions, and that (Wright 1985, p 1795) "each defendant's one unit was necessary for the sufficiency of a set of actual antecedent conditions that included only four of the other units, and the sufficiency of this particular set of actual antecedent conditions was not affected by the existence of two additional duplicative units."

In fact, in this sense, for each defendant's discharge there are fifteen distinct actually sufficient sets of antecedent conditions, one for each possible choice of any four of the 6 remaining defendant's units of pollution.

The causal model for this example has variables X_i, $i=1,\ldots,7$ $i=1,\ldots,7$, representing whether defendant i contributed his one unit of pollution $(X_i=1)$ or not $(X_i=0)$. The single structural equation is

$$\begin{aligned}DP=(&X_1=1\wedge X_2=1\wedge X_3=1\wedge X_4=1\wedge X_5=1)\vee\ldots\vee\\&(X_7=1\wedge X_6=1\wedge X_5=1\wedge X_4=1\wedge X_3=1)\end{aligned}.$$

It consists of 21 terms where each term is a conjunction of 5 of the 7 literals $X_i=1$. Since each literal $X_i=1$ is satisfied in the given scenario $(M,\vec{u})$, each literal occurs in 15 satisfied terms in conjunction with 4 of the remaining 6 X_i or, equivalently, each literal $X_i=1$ occurs in 15 terms without conjuncts involving 2 of the remaining 6 variables. Thus, for any $X_i=1$ and variables X_k, X_l $(i\neq k\neq l)$, there exists some term T_{DP} with $coin_{\vec{u}}^{T_{DP}}(DP;X_i\mid X_k,X_l)$.

Without loss of generality, to show that each defendant's pollution discharge is an actual cause of $DP=1$, let $i=1$ and choose T_{DP} so that $coin_{\vec{u}}^{T_{DP}}(DP;X_1\mid X_6,X_7)$ (i.e., $T_{DP}=(X_1=1\wedge X_2=1\wedge X_3=1\wedge X_4=1\wedge X_5=1)$). Then with $\vec{R}=\langle X_1,DP\rangle$ and $\vec{T}=\langle DP\rangle$, the equation for *DP* in $M_{[\vec{R},\vec{u}]}^{\vec{T}}$ is

$$\begin{aligned}DP=&(X_1=1)\vee(X_6=1)\vee(X_7=1)\vee(X_1=1\wedge X_6=1)\vee(X_1=1\wedge X_7=1)\\&\vee(X_6=1\wedge X_7=1)\vee(X_1=1\wedge X_6=1\wedge X_7=1)\end{aligned}.$$

Since, in $M_{[\vec{R},\vec{u}]}^{\vec{T}}$, DP is a trivial function of the variables in $\{X_2,\ldots,X_5\}$, for $\vec{W}=\mathcal{V}-\vec{R}=\{X_i\}$, $i=2,\ldots,7$, of condition AC2 of the new definition, only the set-

[1] For example, Wright (2001, p 1100) cites the case of Warren v. Parkhurst, 92 N.Y.S. 725 (N.Y. Sup. Ct. 1904), *aff'd*, 93 N.Y.S. 1009 (A.D.1905), *aff'd*, 78 N.E. 579 (N.Y. 1906), where each of twenty-six defendants discharged "nominal" amounts of sewage into a creek which individually were not sufficient to destroy the use of downstream plaintiff's property but the stench of the combined discharges was sufficient.

tings for X_6 and X_7 matter. Setting $(X_6, X_7) = (0,0)$, $X_1 = 1$ is easily seen to satisfy the counterfactual test of condition AC2.

Note that $ACP^{\vec{T}DP}_{[\vec{R},\vec{u}]} = \{X_1, X_2, \ldots, X_5, DP\}$ and, in the causal world $(M, \vec{u})$, $X_1 = 1$ is necessary for the sufficiency of the set including defendant one's discharge ($X_1 = 1$) and only four other discharges.

7.3 Double Omission Cases

A class of cases that have proved problematic for the NESS test, the so-called double omission cases, suggest that modeling is an important aspect of a NESS enquiry in practice: "Some of the most difficult overdetermined-causation cases, at least conceptually, are those involving multiple omissions, which usually involve failures to attempt to use missing or defective safety devices or failures to attempt to read or heed missing or defective instructions or warnings." (Wright 2001, pp. 1123-1124).

Wright (1985, p. 1801; 2001, p. 1124 ff.) considers in detail the case of *Saunders System Birmingham Co. v. Adams*[2] where a car rental company negligently failed to discover or repair bad brakes before renting a car out. The driver who rented the car then negligently failed to apply the brakes and struck a pedestrian. In general, courts have held that individuals who negligently fail to repair a device (or provide proper safeguards or warnings) are not responsible when (negligently) no attempt was made to use the device (or use the safeguards or observe the warnings). According to Wright (2001, p. 1124), the court's decisions reflect a "tacit understanding of empirical causation in such situations": not providing or repairing a device (or not providing proper safeguards or warnings) can have no causal effect when no attempt was or would have been made to use the device (or use the safeguard or observe the warning)—unless no attempt was made because it was known that the device was inoperative (or the safeguards or warnings were inadequate).

Wright's (1985, p. 1801) NESS analysis (where *D* represents the driver, *C* represents the car rental company, and *P* represents the pedestrian) is as follows: *D*'s negligence was a preemptive cause of *P*'s injury. *C*'s negligence did not contribute to the injury. *D*'s not applying the brakes was necessary for the sufficiency of a set of actual antecedent conditions not including *C*'s failure to repair the brakes. The sufficiency of this set was not affected by *C*'s failure to repair the brakes: "A failure to try to use brakes will have a negative causal effect whether or not the brakes are defective." *C*'s failure to repair the brakes was not a necessary element of any set of antecedent actual conditions that was sufficient for the occurrence of the injury: "Defective brakes will have an actual causal effect only if someone tries to use them." The effect of *C*'s failure to repair the brakes was preempted by *D*'s failure to try to use them.

Notice that interchanging *C* and *D*'s negligent acts in this argument results in an apparently equally plausible argument for *C*'s negligence being a preemptive cause of

[2] Saunders Sys. Birmingham Co. v. Adams, 117 So. 72 (Ala. 1928).

P's injury. According to Wright (2001, p.1125): "At the time that I wrote this explanation, I was aware that it was too brief and cryptic, relied upon an insufficiently elaborated notion of causal sufficiency and 'negative causal effect,' and therefore could seemingly be reversed to support the opposite causal conclusions merely by switching the references to the two omissions. Nevertheless, I thought it roughly stated the correct analysis in very abbreviated form."

Binary variables for this causal model might be: *RB* for "repairs brakes", *AB* for "applies brakes", *BO* for "brakes operate", and *HP* for "pedestrian is hit". The question is, what are the structural equations? The structural equations suggested by the argument that in the NESS analysis the roles of *C* an *D* are symmetrical might be $\neg BO = \neg RB \vee \neg AB$ and $PH = \neg BO$.

It is easy to see that the new definition will classify both $\neg RB$ and $\neg AB$ as actual causes of *PH* for this model. On the other hand, suppose the structural equations are $BO = RB \wedge AB$ and $PH = \neg BO$. In that case the new definition will classify neither *RB* nor *AB* as a cause of *PH*. This model captures the intuition that not repairing the brakes is not a cause of the pedestrian being hit if the brakes are not applied but also suggests that not applying the brakes cannot cause the striking of the pedestrian if the brakes are not operative. Notice, however, that in Wright's analysis there is the suggestion of a mechanism that neither of these models includes: not using the brakes will have a causal effect whether or not the brakes are not repaired. In other words, there are two distinct mechanisms for the pedestrian being hit; confusion arises because not braking just happens to play a part in both. On this latter analysis the causal model has equations $BO = RB \wedge AB$ and $PH = \neg BO \vee \neg AB$. The causal diagram for this model is represented in Figure 3.

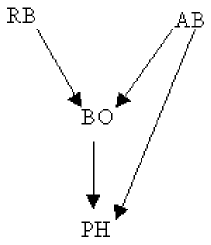


Fig. 3. Causal diagram for the braking scenario

Indeed, for this model the new definition will classify $\neg AB$ as a cause of *PH* but not $\neg RB$. It is this missing mechanism that lies behind the intuitive and analytic confusion in the double omission cases. Wright's initial NESS argument was not incorrect but only lacked an adequate language to represent the causal dynamics of the scenario.

8 Conclusions

Developing a definition of actual or token causation conforming to intuitive judgments is an active problem for both legal scholarship and AI. As an element in the determination of legal responsibility, courts have been required to develop a practical test for actual causation. The accepted test, the ‘but-for’ test, is limited in its application. Wright’s NESS test appears to successfully address these limitations. The NESS test itself requires a counterfactual test. The language of structural models allows for the formal representation of counterfactual arguments. We have presented a formal definition of actual causation in the language of structural models, which we believe captures the essential meaning of the NESS test while successfully avoiding the weaknesses inherent an earlier structural definition of Halpern and Pearl.

Acknowledgements. This research was supported by the second author’s NSERC Discovery Grant.

References

Ashley, Kevin D. (1990). Modeling legal arguments: reasoning with cases and hypotheticals. Cambridge : MIT Press, 329 pages

Baldwin, Richard (2003). A Structural Model Interpretation of Wright’s NESS test. Department of Computer Science, University of Saskatchewan, MSc thesis [Online]. Available: http://library.usask.ca/theses/available/etd-09152003-154313/ [Accessed 2003, Nov. 28].

Baldwin, R., and Neufeld, E. (2003) On the Structure Model Interpretation of Wright’s NESS Test. In *Proceedings of AI 2003,* Halifax (June 2003) LNAI 2671 9-23

Wright, R.W. (1985). Causation in tort law. *California Law Review*, 73, pp. 1735-1828.

Halpern, J.Y., and Pearl, J. (2000). Causes and explanations: a structural-model approach. Retrieved September 3, 2001 from http://www.cs.cornell.edu/home/halpern/topics.html#rau (Part I, Causes, appears in *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA: Morgan Kaufmann, 194-202, 2001.)

Hart, H.L.A., and Honoré, A.M. (1985). *Causation in the law* (2nd ed.). Oxford University Press.

Hopkins, M., and Pearl, J. (2003). Clarifying the Usage of Structural Models for Commonsense Causal Reasoning. In Proceedings of the 2003 AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning, Stanford University, March 24-26, 2003.

Pearl, J. (1995). Causal diagrams for empirical research. *Biometrika*, 82 (4), 669–710.

Pearl, J. (1998). On the definition of actual cause. Technical Report (no. R-259), Department of Computer Science, University of California, Los Angeles.

Pearl, J. (2000). *Causality: Models, reasoning, and inference*. New York: Cambridge University Press.

Wright, R.W. (1985). Causation in tort law. *California Law Review*, 73, pp. 1735-1828.

Wright, R.W. (1988). Causation, responsibility, risk, probability, naked statistics, and proof: Pruning the bramble bush by clarifying the concepts. *Iowa Law Review*, 73, pp. 1001-1077.

Wright, R.W. (2001). Once more into the bramble bush: Duty, causal contribution, and the extent of legal responsibility [Electronic version]. *Vanderbilt Law Review*, 54 (3), pp. 1071-1132.

Spatio-temporal Reasoning for Vague Regions

Zina M. Ibrahim and Ahmed Y. Tawfik

University of Windsor, 401 Sunset Avenue,
Windsor, Ontario N9B 3P4, Canada
{ibrahim,atawfik}@uwindsor.ca

Abstract. This paper extends a mereotopological theory of spatiotemporal reasoning to vague "egg-yolk" regions. In this extension, the egg and its yolk are allowed to move and change over time. We present a classification of motion classes for vague regions as well as composition tables for reasoning about moving vague regions. We also discuss the formation of scrambled eggs when it becomes impossible to distinguish the yolk from the white and examine how to incorporate temporally and spatially dispersed observations to recover the yolk and white from a scrambled egg. Egg splitting may occur as a result of the recovery process when available information supports multiple egg recovery alternatives. Egg splitting adds another dimension of uncertainty to reasoning with vague regions.

1 Introduction

Spatiotemporal knowledge representations proposed so far include qualitative models of kinematics [14] as well as many spatiotemporal logics [9,11]. These logics augment a temporal logic with a spatial one. For example, temporal modal logic and RCC-8 are combined in [17] to represent and reason about spatiotemporal change.

Muller's theory [11,12,13] describes the relations among spatiotemporal objects via a primitive of connectedness. As humans usually reason with imprecise or incomplete information, the notion of vague regions of space-time has attracted some attention [6]. This paper extends Muller's theory to vague regions using the egg-yolk theory [5] as a base.

Due to the importance of space in AI, lot of work has been put into formulating representations of space in the past decade. Also, because systems in AI usually deal with imprecise, incorrect or incomplete information, qualitative models have been the preferred choice as opposed to quantitative ones. In particular, Qualitative Spatial Representation (QSR) models have flourished and are accepted as the models that take as ontological primitives extended regions of space as opposed to points. One such theory is the Region Connectedness Calculus (RCC), which is a well-known first-order theory that deals with regions in a topological manner [15]. The theory's original intension has been for regions with precise boundaries, or in other words crisp regions of space. However, it was extended to deal with vague regions in [4,5,10] and formed what is known as the egg-yolk theory that represents a vague region of space by two crisp regions, the egg and the yolk. Whereas the yolk represents the objects that definitely belong to the region, the white is for the objects that may or may not belong to the region. The objects outside the egg definitely do not belong to the region.

Qualitative theories of time have also had a rich history starting with Allen's interval logic [1] that has paved the way to reason with time in a qualitative manner.

Due to the richness in qualitative theories of both space and time, Qualitative spatiotemporal reasoning was the next natural step. Work in this direction was also motivated by the need to model motion in a qualitative fashion as the usual quantitative methods are not favored in AI. Work in this direction includes Muller's work [11,12,13] which is a mereotopological theory of space-time inspired by [2]. The theory has as primitives extended regions of space-time; it defines all objects made of this primitive to be occurants and views them as space-time histories. The relations between the regions (histories) are based on the temporal and spatial relationships that hold among them, and a set of axioms supports spatiotemporal inference from spatial and temporal information. The theory combines RCC-8 spatial representation and Allen's interval logic. It further defines motion by examining the relationships between histories at two consecutive time intervals and accordingly identifies classes of motion that may be true as regions move [11].

In this paper, we present the building blocks to extending Muller's theory to vague regions (histories) of space-time using the egg-yolk representation. We also aim at constructing a set of motion classes similar to those of Muller found in [11] for vague regions and utilize composition tables to obtain the possible relations between two (or more) moving vague regions. Our approach is partly inspired by the scrambled-egg theory [6,7] in extending the egg-yolk theory from dealing with spatial regions to dealing with regions of space-time. Guesgen's work on scrambled eggs suggests that as regions start to move, less information becomes available regarding their yolk and white, and hence the parthood between the yolk and the egg can no longer be assumed, and a scrambled egg results. Here we examine the possibility of recovering the egg-yolk structure from a scrambled egg by incorporating fixed-interval object re-identification [16]. Object re-identification uses recent observations to reorganize egg-yolk pairs, thus reducing the uncertainty. More specifically, we identify the weaknesses of the egg-yolk theory in the spatiotemporal domain and suggest a solution for overcoming this weakness.

2 Overview of Muller's Spatiotemporal Theory

The primitive entities in Muller's theory are spatiotemporal regions. The relations among these regions can be spatiotemporal as well as temporal. A spatiotemporal relationship describes the spatial motion of a region over time with respect to a static reference region, while the temporal relationship between two regions describes how the two spatiotemporal evolutions are related in time.

The theory follows a general trend in spatiotemporal representation by combining a spatial theory as a topological base, along with a temporal theory and accordingly formulating the spatiotemporal interactions between the two. The RCC-8 set is chosen as the topological base due to its wide acceptance and inherit conceptual neighborhood structure. The temporal relations are those of Allen [1]. The resulting theory is mereotopological and is based on assumptions compatible with RCC-8. The theory does not perceive points as primitives and relies on intervals/regions in representing spatial/temporal quantities. The choice of Allen's interval logic and RCC-8 produces a coherent spatiotemporal theory from this perspective.

The spatial extents of two regions x and y are connected $(C\,xy)$ if the distance between them is zero. The same is also true for the temporal connection between two intervals a and b denoted by $(a\Diamond b)$. The axiom given in 1 captures the relationship between spatiotemporal connectedness and temporal connectedness.

$$C\,xy \rightarrow x \Diamond y \; . \tag{1}$$

Note that for notational clarity we adopt the following convention:

– Spatiotemporal operators precede the regions they operate upon (prefix notation)
– Temporal operators come between the regions they operate upon (infix notation)

The axiom given in(1) is the base for all other types of interaction between the topological (spatial) and temporal aspects of the theory. Another important concept in Muller's theory is the notion of a 'temporal slice' denoted by $(x|y)$, which refers to the subinterval of x corresponding to the "lifetime" of y when $y \subseteq_t x$ [11].

$$x \mid y \text{ is a continuous section of the trajectory of region}\, x \text{ such that } y \subseteq_t x \; . \tag{2}$$

TEMP_IN xy describes the notion of temporary parthood. The following defines the meaning of region x being a part of region y temporarily.

$$TEMP\_IN\,xy \;\text{ if }\; \exists z, u : (z|x \wedge Pzy) \wedge (u|x \wedge (u \subseteq_t y) \wedge \neg Cuy) \; . \tag{3}$$

Classes of motion are identified and formally defined according to the possible interactions between spatiotemporal objects. For example, region x is said to have reached region y during an interval z if x finishes y and x is temporarily in y.

$$REACH\,zxy \;\text{ if }\; TEMP\_IN\,x|z\,y|z \wedge (x|z\,.\,y|z \;FINISHES\; z) \; . \tag{4}$$

In (4), FINISHES is based on Allen's interval logic and is defined as:

$$x\,FINISHES\,y \;\text{ if }\; x \subseteq_t y \wedge \forall z(x < z \rightarrow y < z) \wedge \neg y \subseteq_t x \; . \tag{5}$$

Where $<$ represents a temporal precedence relationship.

The symmetric motion class LEAVE is obtained by replacing FINISHES by STARTS, defined in (6) to finally have the definition given in (7).

$$x\,STARTS\,y \text{ if } x \subseteq_t y \wedge \forall z(z < x \rightarrow z < y) \wedge \neg y \subseteq_t x \ . \tag{6}$$

$$LEAVE\,zxy \text{ if } TEMP\_IN\,x|z\,y|z \wedge (x|z\,.\,y|z\ STARTS\ z) \ . \tag{7}$$

Note that this definition of LEAVE assumes that the smaller region leaves the larger region and cannot be used to capture statements like "Leaving a book" but works for statements like "Leaving town".

The motion of region x is said to be internal in region y during interval z if all time slices of x are spatial proper parts of corresponding time slices of region y throughout the interval z.

$$INTERNAL\,zxy \text{ if } PP\,x|z\ y|z \ . \tag{8}$$

The motion of region x is external to region y throughout the interval z if at no time during z has x been connected to y.

$$EXTERNAL\,zxy \text{ if } \neg C\,x|z\ y|z \ . \tag{9}$$

Region x is considered to have hit region y if the two regions became externally connected at the end of interval z.

$$\begin{gathered} HIT\,zxy \text{ if } EC\,x|z\,y|z \ \wedge\ \forall x_1, y_1[(Px_1\,x|z \wedge Py_1\,y|z \wedge ECx_1y_1) \rightarrow \\ x_1\,FINISHES\,z \wedge y_1\,FINISHES\,z] \ . \end{gathered} \tag{10}$$

Region x may also split into two subregions during the interval z

$$\begin{gathered} SPLIT\,zxy \text{ if } EC\,x|z\,y|z \ \wedge\ \forall x_1, y_1[(Px_1x|z \wedge Py_1y|z \wedge ECx_1y_1) \rightarrow \\ x_1\,STARTS\,z \wedge y_1\,STARTS\,z] \ . \end{gathered} \tag{11}$$

We can use the above motion classes to defined more complex classes. For example, a region x has crossed another region y during an interval z if x reaches y and leaves it in two consecutive subintervals of z.

$$\begin{gathered} CROSS\,zxy \text{ if } \exists\, z_1, z_2(z = z_1 + z_2 \ \wedge\ z_1\,MEETS\,z_2 \wedge\ REACH\,z_1xy \\ \wedge\, LEAVE\,z_2xy) \ . \end{gathered} \tag{12}$$

Where:

$$x\,MEETS\,y \text{ if } x < y \wedge x \diamondsuit y \ . \tag{13}$$

3 Extending Muller's Theory to Vague Egg-Yolk Regions

One of the limitations of Muller's theory as presented in section 2 is that it assumes that the spatiotemporal regions have clearly defined boundaries in both time and space. In many practical situations, it is not possible to represent crisp boundaries for regions in space. For example, it is not possible to precisely represent the boundaries of an ocean, or a mountain. The boundaries of a city or even a property can also be unclear or unknown at a high resolution. To reason about a spatiotemporal mobile region, it is completely unreasonable to assume crisp deterministic motion unless the region's location is continuously and accurately monitored. In spatiotemporal reasoning, spatial and temporal vagueness pervade.

The egg-yolk theory [5] has been introduced to handle spatial vagueness. According to this theory, a region in space with vague boundaries is defined in terms of two crisp regions the egg and the yolk. The vague boundaries may lies anywhere in the egg-white between the yolk and the shell of the egg. The yolk represents the region in space that definitely belongs to the region and any region in space outside the egg is certainly outside the region. Figure 1 shows all the 46 possible topological relations between two egg-yolk regions.

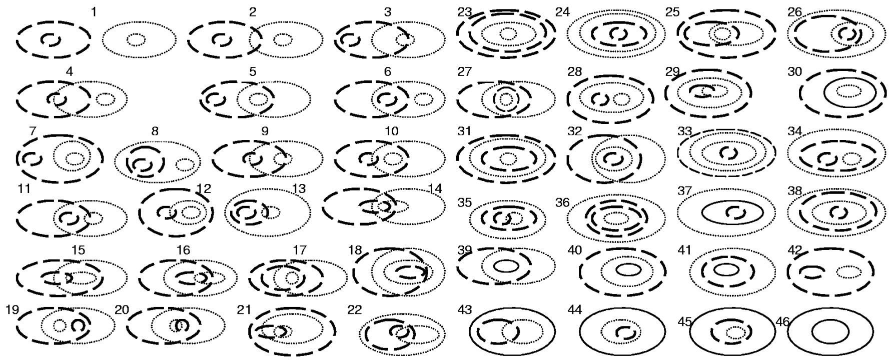


Fig. 1. The 46 Egg-Yolk Relationships

Some of the spatial relationships defined for regions with crisp boundaries in RCC-8 are no longer valid for egg-yolk region. For example, it is not generally correct to describe two egg-yolk regions as externally connected (EC) or tangential proper part (TPP) of one another. [4] has presented a set of five spatial relationships (RCC-5) that may hold between two spatial regions including egg-yolk regions. Given two crisp regions x and y with a set of RCC-8 relations holding between them, if x and y become vague, is it now possible to deduce the set of egg-yolk relations that hold between the vague versions of the two regions using the RCC-5 set shown in Figure 2 along with the RCC-8 set.

3.1 Mapping RCC-8 to RCC-5

Here, we axiomatize a conversion process used in [4] to obtain RCC-5 from RCC-8 and back, as shown in figure 2.

The axiom given in (14) states that regions x and y are distinct regions (non-overlapping) if they are disconnected or externally connected.

$$DRxy = DCxy \vee ECxy \ . \tag{14}$$

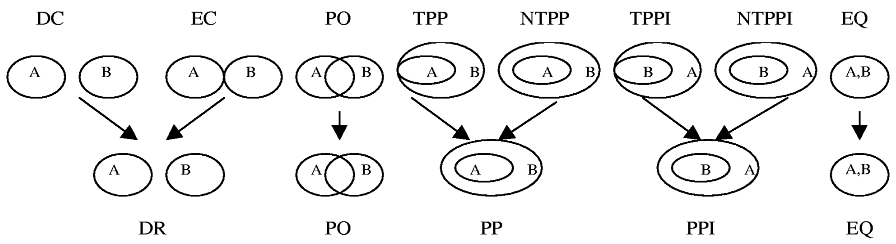


Fig. 2. The RCC-8 and RCC-5 Relations

The axiom given in (15) states that region x is a proper part of region y if it is a tangent proper part or a non-tangent proper part.

$$PP\,xy\ =\ TPP\,xy \vee\ NTPP\,xy\ . \tag{15}$$

The axiom given in (16) provides the inverse relationships to the previous axiom.

$$PPI\,xy\ =\ TPPI\,xy \vee\ NTPPI\,xy\ . \tag{16}$$

The relations EC, TPP and TPPI are based on the exact location of the region boundaries and therefore cannot be defined for a region with vague boundaries. Hence, the exioms given in (17-19) describe impossible situations.

$$\neg\exists x, y\,EC\,xy \wedge\ (VAGUE\,x\ \wedge\ VAGUE\,y)\ . \tag{17}$$

$$\neg\exists x, y\,TPP\,xy \wedge\ (VAGUE\,x\ \wedge\ VAGUE\,y)\ . \tag{18}$$

$$\neg\exists x, y\,TPPI\,xy \wedge\ (VAGUE\,x\ \wedge\ VAGUE\,y)\ . \tag{19}$$

As a result, the relations EC, TPP, TPPI do not have vague counterparts. Given that x and y are vague regions, axioms given in (14-16) become:

$$DR\,xy\ =\ DC\,xy\ . \tag{20}$$

$$PP\,xy\ =\ NTPP\,xy\ . \tag{21}$$

$$PPI\,xy\ =\ NTPPI\,xy\ . \tag{22}$$

3.2 Clustering the EGG-YOLK Relations

The egg-yolk relations shown in Figure 1, which is taken from [5] can be clustered according to the RCC-5 relation that hold between their outer boundaries (boundaries of their eggs) as given in Table 1. The table provides a direct method for finding the relations between the vague versions of two crisp regions x and y having a set of RCC-5 relations holding between them. We add the subscript v to the name of the RCC-5 relations to emphasize that the name given is merely a representation of the possible Egg-Yolk relations that share a common RCC-5 relation between their eggs.

Table 1. Clustering the Egg-yolk Relations

RCC-5(x,y)	Egg-Yolk Group
DR_v	1
PO_v	2 3 4 5 6 9 10 11 14 15 17 18 19 20 27 32 39
EQ_v	42 43 44 45 46
PP_v	8 13 22 24 26 34 35 36 37 38 41
PPI_v	7 12 21 23 25 28 29 30 31 33

As a side note, Figure 1 supports the correctness of the axioms given in (20), (21) and (22). For example, examining the egg-yolk relations that correspond to the RCC-5 relation DR, there is only one case. In this case, the eggs are disconnected (DC) and no case of the eggs related via EC, which satisfies (20). The same reasoning can be used to verify (21) with PP and (22) with PPI.

General Algorithm

The transformations built in the prvious discussions can be formulated as an algorithm that, given two regions x and y and the RCC-8 relations that hold between them, generates a list of all possible egg-yolk pairs given the RCC-8 list.

Algorithm Convert

```
Input: R8 (RCC-8 set)
Output: EY(egg-yolk set)
Var: R5 (list that will hold the RCC-5)
     FinalSet(list of all egg-yolk relations that may hold)
Procedure:
R5 <-- convertToRCC5(R8)
Foreach e in R5 do:
     FinalSet +=ObtainCorrespondingEggYolk(e);
End for
Return (toSet(FinalSet));
End Procedure


Procedure convertToRCC5
input: R8
var: R5
Procedure:
     For each e in R8
              If e == 'DC' R5 += DR
              Else If e == 'PO' R5 += PO
              Else If e == 'NTPP' R5 += NTPP
              Else If e == 'NTPPI' R5 += NTPPI
              Else if e == 'EQ' R5 += EQ
```

```
    End for
    Return (toList(R5));
End Procedure
```

3.3 Motion and Vague Regions

[13] defines the following classes of motion: LEAVE, REACH, CROSS, HIT, INTERNAL, EXTERNAL, and SPLIT. Constructing motion classes for vague regions is different as the class definition must not rely on the fact that the regions have well-defined boundaries. However, by assuming that we have enough information to discern the egg and the yolk for every subinterval throughout the motion, it is possible to redefine the classes of motion for vague regions. This assumption is equivalent to defining a vague time slice denoted by x/y as a continuous portion of the trajectory of region x that occurs during the lifetime of region y provided that yolk x, white x, yolk y and white y are distinct regions throughout the time slice.

$$\begin{gathered} x/y\, is\, continuous\, section\, of\, the\, trajectory\, of\, vague\, region\, x\, such\, that \\ x \subseteq_t y \wedge DR\, yolk\, x\, white\, x \wedge DR\, yolk\, y\, white\, y\, throughout\, x/y\ . \end{gathered} \tag{23}$$

Using the definition for the vague time slice, we can define classes of motion for LEAVE, REACH, CROSS, INTERNAL and EXTERNAL. As the definitions provided in Section 2 for these classes of motion do not rely on knowing the exact boundaries of the region, we can retain the same definitions after restricting them as follows.

$$REACH_v\, z\, x\, y \;\text{ if }\; REACH z\, x\, yolk\, y\ . \tag{24}$$

$$LEAVE_v\, z\, x\, y \;\text{ if }\; LEAVE\, z\, x\, yolk\, y\ . \tag{25}$$

$$INTERNAL_v\, z\, x\, y \;\text{ if }\; INTERNAL\, z\, x\, yolk\, y\ . \tag{26}$$

$$EXTERNAL_v\, z\, x\, y \;\text{ if }\; EXTERNAL\, z\, x\, yolk\, y\ . \tag{27}$$

$$CROSS_v\, z\, x\, y \;\text{ if }\; CROSS\, z\, x\, yolk\, y\ . \tag{28}$$

Where x and y denote the whole region including the vague boundaries, yolk y denotes the region definitely inside the vague boundaries and white y is the vague part of the region.

However, a HIT is not possible to define for vague regions as the intuitive definition relies on external connection between the outer boundaries of the regions in motion. This is obviously not possible for vague regions as their outer boundaries are fuzzy. It is possible to construct an alternative definition suitable for vague regions. We define a POSSIBLE_HIT between two vague regions as the motion class that holds when the yolk part of the two regions overlap.

$$\begin{gathered} POSSIBLE\_HIT\, zxy \;\text{ if }\; PO_v\, yolk\, x|z\, yolk\, y|z \wedge \forall\, x_1, y_1\, [(Px_1\, yolk\, x|z \\ \wedge Py_1\, yolk\, y|z \wedge PO_v x_1 y_1) \rightarrow x_1\, FINISHES\, z \wedge y_1\, FINISHES\, z]\ . \end{gathered} \tag{29}$$

The effect of vagueness is not confined to HIT. The vagueness of the boundaries may necessitate that we define a POSSIBLE_REACH to describe the situation when region x reaches the white of region y.

$$POSSIBLE\_REACH\,zxy \text{ if } REACH\,zx\,white\,y\ . \tag{30}$$

Similarly, a POSSIBLE_LEAVE occurs when region x departs from the white of region y.

The motion of region x is POSSIBLE_INTERNAL to region y if it is internal to the egg but not confined to the yolk of region y.

$$\begin{gathered}POSSIBLE\_INTERNAL\,zxy \text{ if } INTERNAL\,zxy \wedge \\ \neg INTERNAL\,zx\,yolk\,y\ .\end{gathered} \tag{31}$$

The motion of region x is POSSIBLE_EXTERNAL to region y if it is external to yolk y but not to the whole egg y.

$$\begin{gathered}POSSIBLE\_EXTERNAL\,zxy \text{ if } EXTERNAL\,zx\,yolk\,y \wedge \\ \neg\, EXTERNAL\,zxy\ .\end{gathered} \tag{32}$$

Region x is considered to have possibly crossed region y if it had possibly reached it and possibly left it in two consecutive time intervals.

$$\begin{gathered}POSSIBLE\_CROSS\,zxy \text{ if } \exists\, z_1, z_2\,(z = z_1\, + z_2) \wedge z_1\, MEETS\, z_2 \\ \wedge\, POSSIBLE\_REACH\, z_1xy \wedge POSSIBLE\_LEAVE\, z_2xy\ .\end{gathered} \tag{33}$$

We have intentionally left the discussion of SPLIT to Section 5 as it is more appropriate to examine it along with the issues related to scrambled eggs.

4 Reasoning about Motion

Knowing that two spatiotemporal regions u and w are moving with respect to one another according to one of the motion classes during a certain temporal interval allows us to infer a set of spatial relationships that may hold during another interval. For example, knowing that region u is leaving region w during interval z would allow us to conclude that one of the following three spatial relationship may hold during an interval y that overlaps the interval z as shown in figure 3:

- Object u starts off as a part of w *(PP_v uw)*
- Object u can still partially overlap w *(PO_v uw)*
- Object u can be completely outside w *(DR_v uw)*.

Therefore the temporally ordered spatial sequence [PP$_v$ *uw*, PO$_v$ *uw*, DR$_v$ *uw*] corresponds to a LEAVE$_v$ motion. As REACH$_v$ is the opposite of LEAVE$_v$, the temporally ordered sequence describing REACH$_v$ is [DR$_v$ *uw*, PO$_v$ *uw*, PP$_v$ *uw*].

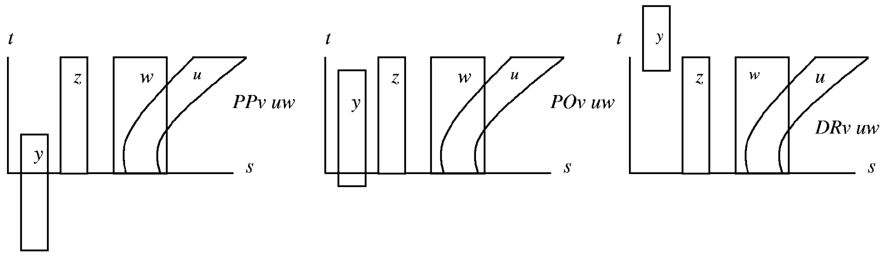


Fig. 3. The Three Possibilities for LEAVE

Similarly, a POSSIBLE_HIT can be described by the sequence [DR$_v$ uw, PO$_v$ uw]. Other motion classes such as INTERNAL$_v$ and EXTERNAL$_v$ can be captured by the singleton spatial relationship [PP$_v$ uw] and [DR$_v$] respectively. CROSS can be obtained by concatenating the sequences for REACH$_v$ and LEAVE$_v$ to obtain [DR$_v$ uw, PO$_v$ uw, PP$_v$ uw, PO$_v$ uw, DR$_v$ uw]. The sequences for POSSIBLE_LEAVE, POSSIBLE_REACH, POSSIBLE_INTERNAL, and POSSIBLE_EXTERNAL are identical to the sequences for LEAVE$_v$, REACH$_v$, INTERNAL$_v$, and EXTERNAL$_v$ respectively.

Frequently, we have information describing what happens over a time interval and we want to make deduction relative to another interval. For example, if we know that the motion of u with respect to w during z is INTERNAL$_v$ zuw, then it is possible to deduce INTERNAL zuw during any x DURING z. If instead we are given LEAVE$_v$ zuw, then our conclusion becomes INTERNAL$_v$ xuw $\vee$ LEAVE$_v$ $xuw \vee$ EXTERNAL$_v$ xuw. In general we cannot make conclusions about intervals before or after the motion.

To generalize, assume that a temporal relationship R$_t$ holds between intervals x and z, and that region u makes a motion of class M relative to region v over interval z, what spatial relationships hold over interval x? Table 2 answers this question for the classes of motions we discussed and eleven of Allen's thirteen relationships as it is not possible to make any conclusions for BEFORE and AFTER. Note that POSSIBLE_LEAVE, POSSIBLE_REACH, POSSIBLE_INTERNAL, and POSSIBLE_EXTERNAL behave in the same fashion as LEAVE$_v$, REACH$_v$, INTERNAL$_v$, and EXTERNAL$_v$ respectively.

Table 3 on the other hand, answers a different question. Assuming that region u is moving with respect to region v and that region v has a static spatial relation R$_s$ with respect to region w, then what can be deduce about region u with respect to region w.

5 Scrambled Eggs and Region Splitting

Section 4 assumes that it is possible to distinguish the yolk from the white at all times. This assumption is rather unrealistic in some situations. As the object starts moving, it often becomes impossible to distinguish the white from the yolk.

Table 2. Composing Motion with Temporal Information

Motion R_t	$LEAVE_v$	$POSSIBLE\_HIT_v$	$REACH_v$	$CROSS_v$	$INTERNAL_v$	$EXTERNAL_v$
MEET	PO_v, PP_v	DR_v, PO_v	DR_v, PO_v	DR_v, PO_v	PP_v, PO_v	DR_v, PO_v
Ot	PO_v, PP_v	DR_v, PO_v	DR_v, PO_v	DR_v, PO_v	PP_v, PO_v	DR_v, PO_v
START	PO_v, PP_v	DR_v	DR_v, PO_v	DR_v, PO_v	PP_v	DR_v
DURING	All	DR_v	All	All	PP_v	DR_v
=t	PO_v	PO_v	PO_v	PO_v	PP_v	DR_v
FINISH	DR_v, PO_v	PO_v	PP_v, PO_v	DR_v, PO_v	PP_v	DR_v
$MEET_i$	DR_v, PO_v	PO_v	PP_v, PO_v	DR_v, PO_v	PO_v, PP_v	DR_v, PO_v
$FINISH_i$	PO_v	PO_v	PO_v	PO_v	PO_v, PP_v	DR_v, PO_v
$START_i$	PO_v	PO_v	PO_v	PO_v	PO_v, PP_v	DR_v, PO_v
$DURING_i$	PO_v	PO_v	PO_v	PO_v	PO_v, PP_v	DR_v, PO_v

Table 3. Composing Motion with Spatial Information

Motion R_s	$LEAVE_v$	$POSSIBLE\_HIT_v$	$REACH_v$	$CROSS_v$	$INTERNAL_v$	$EXTERNAL_v$
DR_v	DR_v, PO_v	DR_v, PO_v, PPI_v	DR_v, PO_v	DR_v, PO_v	DR_v	All
PO_v	DR_v, PO_v, PP_v	DR_v, PO_v, PP_v	DR_v, PO_v, PP_v	All	All	All
PP_v	PP_v, PO_v	PP_v, PO_v	PP_v, PO_v	PP_v, PO_v	PP_v	All
PPI_v	DR_v, PO_v	PO_v, DR_v	DR_v, PO_v	DR_v, PO_v	All	DRv
$=_v$	PO_v	PO_v	PO_v	PO_v	PP_v	DR_v

Initially, we may have enough information regarding the yolk and egg making up the region, but as a vague region starts moving, the degree of certainty of information at hand decreases and the range of the egg and the yolk start expanding. With time, the uncertainty reaches a level where distinguishing the yolk from the white becomes impossible and we end up with a scrambled egg [6, 7]. Usually, there is high probability that the object is located in a relatively small subregion of the scrambled egg. [6] uses fuzzy membership functions to represent the possibility of locating the object within the scrambled egg. For example if Tom leaves home at 8:00 AM, at 9:00 AM he may be located anywhere in a scrambled egg that may extend 100 kms in all directions. It is much more likely however that he is still in town and even a higher probability that he is at work.

Here, instead of taking a probabilistic approach to the problem, we propose a logic approach. This approach is nonmonotonic as it allows considering multiple possible scenarios concurrently and prefers the one best supported by available evidence. To illustrate this approach, let us consider a slightly modified version of the example in the previous paragraph.

- Tom left at 8:00 AM.
- Tom could have gone to the office.
- Tom could have gone to meet a client.
- The office called at 9:00 A.M. Tom is not in the office.

The first statement is enough to form a first scenario: a scrambled egg that is growing as a function of time in all directions at the fastest speed we could expect Tom to travel at. This is the fundamental scenario and any other scenario must be spatially constrained to the egg defined by the fundamental scenario. The second statement allows a second scenario that confines Tom to his office. This scenario cannot be valid except from the time 8:00 + Δt1, where Δt1 represents the time necessary for the scrambled egg in the fundamental scenario to reach the office. The third statement licenses us to create a third scenario placing Tom at the client's site from 8:00 + Δt2 where Δt2 is the shortest possible time to reach the client's site. The fourth statement introduces a conflict with the second scenario, forcing its retraction. Now we are left with the fundamental (first) and third scenarios. The fundamental scenario will continue to grow until we get reliable and conclusive evidence regarding Tom's location at a particular time forcing the fundamental scenario to shrink and start expanding over time again. All scenarios other than the fundamental one use persistence and causation. The fundamental scenario provides a general consistency constraint and a failsafe answer that may not be very useful.

In [16] a simplified version of this approach has been implemented to re-identify vehicles as they travel along the highway. Each highway section is equipped with inductive loop detectors that record vehicle length, lane, speed and electromagnetic signature. By forming the fundamental eggs and matching signature, length, and lane information, a large majority of vehicles has been correctly re-identified. In many cases several matches resulted for the same vehicle. In these cases, all the possible matches are kept as mutually exclusive possible scenarios.

Sometimes, it is possible to justify that the set of mutually exclusive scenarios is exhaustive (i.e. it is not possible that no new information will justify a new scenario). In these cases, the fundamental scenario is not needed. In these cases, we consider that a scrambled egg SPLIT has occurred. This scrambled egg split is very different from the SPLIT of a crisp region as defined in (11) because the regions a scrambled egg splits into do not have to be continuous, and they do not have to add up to form the original egg. Splitting a non-scrambled egg-yolk region does not represent the same problem as a scrambled egg. In this case, the egg must be split the same way as for a crisp region according to 11. Note that splitting is the only case where it is possible to use EC with non-scrambled egg-yolk regions. As we split a vague region into two subregions, the subregions are externally connected irrespective of the boundaries. A scrambled egg region however, may split into two or more egg-yolk pairs depending upon the information obtained from the object re-identification as one or more objects may satisfy the characteristics of our object in motion. This can be problematic as re-identification splits our scrambled-egg into two objects; this implies allowing branching time which is strictly prohibited by the continuity assumptions guiding this work. We overcome this difficulty by placing each region resulting from the splitting in a separate scenario. Within each scenario, time is linear and motion of all regions is continuous. The only way to combine information from different scenarios is through the use of disjunctions.

6 Conclusion and Ongoing Research

We have found a methodology to modify Muller's representation crisp spatiotemporal regions to accommodate vague regions. We applied this methodology to perform reasoning using the composition tables to answer queries to obtain spatial and temporal information regarding two moving regions. We have also found a subset of the motion classes defined by Muller to suit vague regions. Current research is taking into account the composition tables and formulating general rules to reduce the size of the result of the compositions by trying to eliminate some of the results. Also, we are currently trying to answer different queries by varying the entries in the composition tables to suit the query at hand. Another open question is how to formalize the object re-identification and incorporate it into this first-order theory.

References

1. Allen, J.: Maintaining knowledge about temporal intervals. Communications of the ACM. **26** (1983) 832-843
2. Asher, N. and Vieu, L.: Towards a geometry of commonsense: A semantics and a complete axiomatisation of mereotopology. In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI'95)(1995)
3. Clarke, B.: A calculus of individuals based on 'connection'. Notre Dame Journal of Formal Logic. **22** (1981) 204-218
4. Cohn, N. and Gotts, N.: A theory of spatial regions with indeterminate boundaries. C. Eschenbach, C. Habel and B. Smith (eds), Topological Foundations of Cognitive Science 1994
5. Cohn, N. and Gotts, N.: The 'Egg-Yolk' representation of regions with indeterminate boundaries. In Burough, P. and Frank, A., eds. Geographical Objects with Indeterminate Boundaries, GISDATA **2**(1996) 171-187.
6. Guesgen, H.: From the Egg-Yolk to the scrambled-egg theory. Proceedings of the Fifteenth Internatinal Florida Artificial Intelligence Research Symbosium (FLAIRS'2002) (2002)
7. Guesgen, H.: When regions start to move. Proceedings of the Sixteenth Internatinal Florida Artificial Intelligence Research Symbosium (FLAIRS'2003)(2003)
8. Hazarika, S., and Cohn, A.: Qualitative spatiotemporal continuity. conference on Spatial Information Theory (COSIT) (2001) 92-107.

9. Hornsby, K. and Egenhofer, M.: Identity-based change: A foundation for knowledge representation International Journal of Geographical Information Science. **14** (2000) 207-224
10. Lehmann, F. and Cohn, A.: The EGG/YOLK reliability hierarchy: semantic data integration using sorts with prototypes. Proceedings of Conference on Information Knowledge Management. (1994)
11. Muller, P.: A qualitative theory of motion based on spatiotemporal primitives. Principles of Knowledge Representation and Reasoning (1998).
12. Muller, P.: Space-Time as a primitive for space and motion. Proceedings of FOIS, Frontiers in Artificial Intelligence Applications (1998).
13. Muller, P.: Topological spatiotemporal reasoning and representation. Computational Intelligence **18(3)** (2002) 420-450.
14. Rajagopalan, R. and Kuipers, B.: Qualitative spatial reasoning about objects in motion. Proceedings of the IEEE Conference on Artificial Intelligence for Applications, vol. 46. (1994)
15. Randell, D., Cui, Z., Cohn, A.: A spatial logic based on regions and connection. Proceedings of Knowledge Representation and Reasoning, KR & R **92** (1992) 165-176.
16. Tawfik, A., Peng, A., Tabib, S. and Abdulhai, B.: Learning spatiotemporal context for vehicle reidentification. Proceedings of the 2nd IEEE International Symposium on Signal Processing and Information Technology. (2002)
17. Wolter, F. and Zakharyaschev, M.: Spatiotemporal representation and reasoning based on RCC-8. In Proceedings of the 17th Conf. on Principles of Knowledge Representation and Reasoning, KR-2000. : (2000) 3-14.

Average Case Self-Duality of Monotone Boolean Functions

Daya Ram Gaur[1] and Ramesh Krishnamurti[2]

[1] Department of Math and Computer Science
University of Lethbridge
Lethbridge, AB
Canada, T1K 3M4
gaur@cs.uleth.ca
http://www.cs.uleth.ca/~gaur

[2] School of Computing Science
Simon Fraser University
Burnaby, BC
Canada, V5A 1S6
ramesh@cs.sfu.ca
http://www.cs.sfu.ca/~ramesh

Abstract. The problem of determining whether a monotone boolean function is self-dual has numerous applications in Logic and AI. The applications include theory revision, model-based diagnosis, abductive explanations and learning monotone boolean functions. It is not known whether self-duality of monotone boolean functions can be tested in polynomial time, though a quasi-polynomial time algorithm exists. We describe another quasi-polynomial time algorithm for solving the self-duality problem of monotone boolean functions and analyze its average-case behaviour on a set of randomly generated instances.

Keywords: Knowledge Representation, Machine Learning, Self-duality, Monotone Boolean Functions, Satisfiability.

1 Introduction

The problem of determining if a monotone boolean function in disjunctive normal form containing n clauses is self-dual is ubiquitous. Here we describe an application in the area of Model based diagnosis, in particular the problem of computing all diagnoses is equivalent to determining self-duality. This application is from Eiter and Gottlob [9]. For detailed list of applications in Logic and Artificial Intelligence and Data mining we refer the reader to [10]. For additional applications in the areas of knowledge discovery and data mining we refer the reader to [6].

Diagnostic reasoning, a technique proposed by deKleer and Williams [7] and studied by Reiter [27], attempts to identify the dependencies between components of a model and the discrepancies in the observations. Suppose we have a model for some circuit and the output is not correct for a particular input.

Diagnosis entails identifying the minimal set of components such that failure of any one component in the set could have caused the malfunction. Such a set is called a *conflict set*. A conflict set is minimal if it does not contain any other conflict set. A model can now be described as a disjunction of minimal conflict sets, where each conflict set is a conjunction. For example, let M be a circuit with three components $\{m_1, m_2, m_3\}$. One of the conflict sets could be $\{m_1, m_2\}$ associated with some observation c. This means that if c has been observed then either one of m_1 or m_2 is not functioning correctly. To comment on the nature and properties of conflicts we quote deKleer and Williams from [7].

> For complex domains any single symptom can give rise to a large set of conflicts, including the set of all components in the circuit. To reduce the combinatorics of diagnosis it is essential that the set of conflicts be represented and manipulated concisely. If a set of components is a conflict, then every superset of that set must also be a conflict. Thus the set of conflicts can be represented concisely by only identifying the minimal conflicts, where a conflict is minimal if it has no proper subset which is also a conflict. This observation is central to the performance of our diagnostic procedure. The goal of *conflict recognition* is to identify a complete set of minimal conflicts.

Formally, a *system* is a pair $(SD, COMPONENTS)$ where, SD is a set of first order sentences and $COMPONENTS$ is a finite set of constants. SD contains a unary distinguished predicate called $AB()$ where $AB(x)$ is interpreted to mean that x is 'abnormal'. An *observation* of a system is a finite set of first order sentences denoted OBS.

Definition 1 (Diagnosis) *A diagnosis for* $(SD, COMPONENTS, OBS)$ *is a minimal set* $\Delta \subseteq COMPONENTS$ *such that*

$$SD \cup OBS \cup \{AB(c) | c \in \Delta\} \cup \{\neg AB(c) | c \in COMPONENTS - \Delta\}$$

is consistent.

In other words a diagnosis is the smallest set of components such that the assumption that each abnormal component is in the set together with the assumption that all the other components are not abnormal is consistent with the observations and the system description.

Definition 2 (Conflict set) *A* Conflict set *is a set* $C \subseteq COMPONENTS$ *such that*

$$SD \cup OBS \cup \{\neg AB(c) | c \in C\}$$

is consistent.

We denote by CS the set of all the minimal conflict sets of $(SD, COMPONENTS, OBS)$.

Eiter and Gottlob [9] show the following theorem:

Theorem 1 (Eiter and Gottlob [9]). *$\Delta \subseteq COMPONENTS$ is a diagnosis iff*
$\Delta \in Tr(COMPONENTS, CS)$.

Where Tr is defined as follows:

Definition 3 (Hypergraph) *A hypergraph H is pair (V, E) where V is a finite set and E is a family of finite subsets of V.*

A *transversal* of a hypergraph $H = (V, E)$ is a set $V' \subseteq V$ such that $V' \cap E_i \neq \phi$ for all $E_i \in E$.

Definition 4 (Transversal hypergraph $Tr(H)$) *The transversal hypergraph $Tr(H)$ of a hypergraph H is the family of all minimal transversals of H.*

Next we define the problem of enumeration of the edges of the transversal hypergraph.
PROBLEM: TRANS-ENUM
INSTANCE: Hypergraph $G = (V, E)$.
QUESTION: The edges of the transversal hypergraph $Tr(G)$.

A corollary [10] to Theorem 1. (Eiter and Gottlob [9]) establishes that the problem of computing all diagnoses is $TRANS - ENUM$ complete. Furthermore it is well known that from the point of computability in polynomial time $TRANS - ENUM$ is *equivalent* to self-duality [10].

Next we describe an application in the area of digital signal processing. A class of non-linear filters called stack filters have been proposed by Wendt, Coyle and Lin [28]. Stack filters can be uniquely represented using monotone boolean functions. If the stack filter identifies some vector x as being noise then the complement of x cannot be noise. Hence, it is in the interest of the filter designer to ensure that for every vector and its complement exactly one of them is classified as noise. This problem is equivalent to determining if a monotone boolean function is self-dual.

Suppose we are to come up with a model which classifies the data into 'good' or 'bad' based on some features. Without loss of generality we can assume that the features take on boolean values. A good pattern classifier by definition should be able to classify every input pattern. Also, if an input x is 'good' then its complement cannot be 'good' and vice versa. Determining if a pattern classifier has the above mentioned property is equivalent to determining if some monotone boolean function is self-dual [22].

In this paper we describe another quasi-polynomial time algorithm for the self-duality problem and analyze its average case behaviour on a set of randomly generated instances. The paper is organized as follows. In the next section, we briefly summarize the known results. Section 3. presents the definitions. In Section 4. we describe a quasi polynomial time algorithm for the self-duality problem. Section 5. describes the model used to generate the set of random instances. We analyze the average case behaviour of the algorithm in Section 6. Technical lemmas and theorems are in the appendix.

2 Related Work

The problem of self-duality arises in artificial intelligence [27], databases [25], convex programming [19] and hypergraph theory [9], to name a few. For a detailed list of applications in Logic and AI we refer the reader to the paper by Eiter and Gottlob [10]. The applications include Theory revision, Model-Based Diagnosis [10] and Abductive Explanations [11]. Efficient algorithms for self-duality also imply that monotone boolean functions can be learned efficiently using only membership queries [17,8]. One of the key problems in data mining and knowledge discovery is the generation of frequent and infrequent sets, a problem closely related to the self-duality problem [6]. The exact complexity of determining if a monotone boolean function is self-dual is open. Fredman and Khachiyan [14] provide an $O(n^{4o(\log n)+O(1)})$ algorithm for solving the problem. Bioch and Ibaraki [1] describe a host of problems which are equivalent to determining self-duality. They also address the question of the existence of incremental polynomial algorithms for solving the problem of determining the self-duality of monotone boolean functions. In a related paper [2] they define a decomposition of the problem and give an algorithm to determine a minimal canonical decomposition. Bioch and Ibaraki [3] describe an incremental polynomial algorithm for generating all monotone boolean functions of n variables. It has been shown that for $2-$ *monotone* [5] boolean functions, it is possible to check the self-duality in polynomial time. Bioch and Ibaraki [3] define *almost self-dual* functions as an approximation to the class of self-dual functions. They describe an algorithm based on almost self-duality to determine if a function is self-dual. The complexity of their procedure is exponential in the worst case. Ibaraki and Kameda [20] show that every self-dual function can be decomposed into a set of majority functions over three variables. This characterization in turn gives an algorithm (though not polynomial) for checking self-duality. Makino and Ibaraki [24] define the latency of a monotone boolean function and relate it to the complexity of determining if a function is self-dual. Gaur and Krishnamurti [16] show that the self-duality of monotone boolean functions which have clause sizes bounded by some constant can be determined in linear time and also establish that the problem admits a polynomial time solution under various syntactic restrictions. Recently Eiter, Gottlob and Makino [21] have developed polynomial algorithms and output polynomial algorithms for other classes of monotone boolean functions. They also show that using $\log^2 n$ random bits, self duality can be determined in polynomial time. Makino [23] describes an algorithm to dualize $O(\log n)$ term monotone disjunctive normal forms.

3 Definitions

Given a boolean function $f(x_1, x_2, \ldots, x_n)$, we define its dual denoted by f^d as follows:

Definition 5 *Dual:* $f^d(x) = \bar{f}(\bar{x})$, *for all vectors* $x = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ *where* $\bar{x}$ *is the component wise complement of vector* x.

Next we define monotone boolean functions.

Definition 6 *Monotone boolean function: A boolean function f is monotone if $\forall x, y \in \{0,1\}^n$ $f(x) \leq f(y)$. A vector $x \leq y$ if $x_i \leq y_i,\ i \in \{1..n\}$.*

Equivalently, a boolean function is monotone if it can be represented by an expression which does not contain any negative literals. If a monotone function f is in disjunctive normal form (DNF) then f^d can be obtained by interchanging every *and* operator with an *or* operator and vice versa. f^d is then in conjunctive normal form (CNF). In this paper we are only concerned with monotone boolean functions and sometimes we will refer to them as just boolean functions when the meaning is clear from the context. Self-duality can now be defined as:

Self-duality
INSTANCE: A boolean function $f(x_1, x_2, \ldots, x_n)$.
QUESTION: For every vector $x = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ is $f^d(x) = f(x)$?
From the definition of self-duality it follows that:

Property 1 A boolean function f is self-dual $\iff$ for all vectors $x \in \{0,1\}^n$, $f(x) \neq f(\bar{x})$.

We can assume that the monotone function f is in DNF. Next we show that if there exists a pair of clauses in a monotone function f which do not intersect in any variable, then f is not self-dual. This observation is also implicit in [14].

Lemma 1. *If there exists a pair of non-intersecting clauses in a monotone function f, then f is not self-dual.*

Proof. Let C_1 and C_2 be two such clauses. We construct a vector $x \in \{0,1\}^n$ such that all the variables occurring in C_1 are set to 1 and all the variables occurring in C_2 are set to 0. The remaining variables are arbitrarily set to 0 or 1. $f(x) = 1$ as the clause C_1 evaluates to 1. Also, $f(\bar{x}) = 1$ as C_2 evaluates to 0 on x. Hence by Property 3, f is not self-dual.

Lemma 1 allows us to focus only on those monotone boolean functions in which every pair of clauses intersect. Another property which we use throughout this paper is the following:

Property 2 Every variable in f belongs to at least 2 terms in f.

Property 3 coupled with Lemma 1 implies that each term has at most n variables where n is the total number of clauses in f. Therefore the total number of variables $m \leq n^2$ in f. Given such a function f, we now construct the NAESPI problem and show the equivalence of the two problems. Next we define the NAESPI problem.

NAESPI
INSTANCE: Given a set of variables $V = (v_1, v_2, \ldots, v_m)$, and a collection of clauses $C_i, i = \{1, \ldots, n\}$, $C_i \subseteq V$, every pair of clauses C_i, C_j has a non-empty intersection.

QUESTION: Find a set $S \subseteq V$ such that S contains at least one variable from every clause, but no clause is contained in S.

We are given a monotone boolean function f in DNF form. P is obtained by interpreting the function as a CNF formula. In other words, if $f = (x_1 \wedge x_2) \vee (x_1 \wedge x_3) \vee (x_2 \wedge x_3)$ then $P = (x_1 \vee x_2) \wedge (x_1 \vee x_3) \vee (x_2 \vee x_3)$. Note that every pair of clauses in P intersect since every pair of clauses in f intersect. The next proposition states that the complement of a solution to a given NAESPI problem P is also a solution to P. It was established in [16] that the two problems are equivalent.

Theorem 2 ([16]). *f is not self-dual $\iff$ P is satisfiable.*

4 Algorithm

Let C be an NAESPI problem. We denote the set of variables in C by V and the i^{th} clause by C_i.

Definition 7 (Partial assignment) *A* partial assignment *is a subset $V' \subseteq V$ such that all the elements in V' are assigned a boolean value.*

Given a partial assignment V', we divide the set of clauses C into three sets.

1. P denotes the set of clauses which have at least one variable set to 1 and no variable set to 0. P stands for the clauses with a positive prefix.
2. N denotes the set of clauses which have at least one variable set to 0 and no variable set to 1. N stands for the clauses with a negative prefix.
3. PN is the set of clauses which have at least one variable set to 1 and at least one variable set to 0.

It should be noted that P, N, PN are disjoint sets and the clauses in PN are satisfied as they contain at least one variable set to 1 and at least one variable set to 0.

Definition 8 (Consistent partial assignment) *A partial assignment is said to be* consistent *if it is contained inside some solution to the NAESPI.*

The next lemma shows that for a consistent partial assignment there is a clause either in P or in N which has a 'particular' type of assignment. Next, we define the 'particular' type of assignment.

Definition 9 ($10^*(01^*)$ assignment) *A partial assignment is said to be of type $10^*(01^*)$ if exactly one variable is set to 1 and all the other variables are set to 0 (if exactly one variable is set to 0 and all the other variables are set to 1).*

Lemma 2. *Given sets of clauses P with respect to a consistent partial assignment V', there exists a clause in P which is satisfied as 01^* in the solution containing V'.*

Proof. Let S be the solution and all the clauses in P have at least two variables set to 0. By changing one of the variables to 1 we decrease the number of $0's$ in this clause and no clause is unsatisfied, as clauses in N already have a 0 assigned to them. We iterate until there is one clause with exactly one variable set to 0.

Lemma 3. *Given sets of clauses N with respect to a consistent partial assignment V', there exists a clause in N which is satisfied as 10^* in the solution containing V'.*

Proof. Similar to Lemma 2.

Definition 10 (Unsafe variable in N) *A variable $v \in N$ is said to be* unsafe in N *if there exists a clause $C_v \in N$ containing v such that setting v to* 1 *and all the other variables to* 0 *satisfies at most half the clauses in P.*

Symmetrically we can define unsafe variables in P.

Definition 11 (Unsafe variable in P) *A variable $v \in P$ is said to be* unsafe in P *if there exists a clause $C_v \in P$ containing v such that setting v to* 0 *and all the other variables to* 1 *satisfies at most half the clauses in N.*

Next we describe an $O(n^{2\log n+2})$ algorithm for NAESPI. We begin by trying out all the 10^* possibilities at the top level. For every choice made we compute the sets P, N, PN and denote the resulting subproblem by C_1. Next we compute all the unsafe variables in N (denoted U). Now we generate another subproblem C_2 obtained by collapsing all the variables in U. We solve C_2 recursively by trying out all the 10^* possibilities for clauses in N. We solve C_1 recursively but we only try those 10^* possibilities for which some variable in U is set to 0. Formally,

1. Let $P = N$ be the set of all the clauses and $NP = \phi$.
2. **If** $P = \phi$ **return** satisfiable.
3. Compute U, the set of unsafe variables in N.
4. For every $u \in U$, solve the subproblem (obtained by setting $u = 0$) recursively.
5. If all the subproblems in Step 4 fail then set all the variables in U to 1 and solve all the subproblems (obtained by 10^* assignments) recursively after removing all the satisfied clauses from P and N.
6. If all the subproblems fail at the top most level **then return** unsatisfiable.

Correctness of the algorithm follows from Lemma 2. To prove a bound on the running time we need the following lemmas.

Lemma 4. *If there are no unsafe variables in N(P) then for every $10^*(01^*)$ trial at least half the clause in P(N) are satisfied.*

Proof. Follows from the definition of unsafe variables.

If $N(P)$ contains unsafe variables then we cannot guarantee that every trial of $10^*(01^*)$ in $N(P)$ reduces the number of unsatisfied clauses in $P(N)$ by half. Let U denote the set of all the unsafe variables in $N(P)$. If all the variables in U get assigned a value 1(0) for $U \in N(P)$ then we have satisfied at least one clause in $N(P)$. If one of the variables in U gets assigned 0(1) then at least half the clauses in $P(N)$ are satisfied due to the fact that the variable is an unsafe variable. Let C' be the problem obtained by contracting all the variables in $U \to u$. It should be noted that C' has a solution if and only if all the variables in U get assigned the same value in a solution to C. The next lemma puts a bound on the number of problems and the size of each subproblem generated for C'. Assume that C' was obtained by contracting the variables in N. Let $S(n)$ denote the number of subproblems generated to solve a subproblem comprising n clauses obtained after contracting all the unsafe variables U.

Lemma 5.

$$S(n) \leq n^2 S(\frac{|P|}{2})$$

Proof. The total number of clauses is n and each clause is of size at most n. Hence the total number of 10^* choices after collapsing all the variables in U is at most n^2.

Note that the subproblems obtained do not have any unsafe variables in N. This is true because U is the maximum set of unsafe variables in N. By Lemma 4, the size of P is reduced by half for every subproblem. Hence,

$$S(n) \leq n^2 S(\frac{|P|}{2}) \tag{1}$$

Symmetrically, if U was the set of unsafe variables in P and the subproblems were obtained by contracting all the variables in U then the next lemma holds.

Lemma 6.

$$S(n) \leq n^2 S(\frac{|N|}{2})$$

Proof. Similar to the proof for Lemma 5.

Theorem 3. *The algorithm terminates in $O(n)^{2log(|P|+2)}$ time.*

Proof. Let N and P be the negative and positive sets of clauses in some stage i. Assume without loss of generality that $|P| \leq |N|$. Let U be the set of unsafe variables in N. Assuming the instance is satisfiable, we have the following two cases:

– All the variables in U are assigned a value of 1 in the solution. The collapsed problem $U \to u$ also has a solution. By Lemma 5 this requires no more than

$$S(n) \leq n^2 S(\frac{|P|}{2}) \tag{2}$$

subproblems to be solved.

– Some variable in U is set to value 0 in the solution. In this case we try out at most n^2 possible 10^* assignments and for each such trial at least half the clauses in P are satisfied. This is due to the fact the variable we set to 0 belongs to at least $\frac{|P|}{2}$ clauses (this is implied by the definition of unsafe variables). This requires no more than

$$S(n) \leq n^2 S(\frac{|P|}{2}) \tag{3}$$

subproblems to be solved.

Hence the total number of subproblems generated is at most $O(n^{2\log|P|})$. As it takes $O(n^2)$ time to verify the solution, the total running time is $O(n^{2\log|P|+2})$.

5 Random NAESPI

In this section we examine the average case behaviour of a variant of the algorithm presented in Section 4 on a class of randomly generated NAESPI instances. Considerable research has been conducted in analyzing the average case behaviour of various algorithms for the satisfiability problem. Two models which have been used extensively for generating instances for the satisfiability problem are the constant density model and the constant component model.

According to the *constant density model* ($M_1(n, r, p)$), each of the n clauses is generated independently from the r boolean variables, where each literal belongs to a clause with probability p. In the *constant component model* ($M_2(n, r, k)$), each of the n clauses is selected independently and uniformly from the set L of all k-literal clauses over r variables.

Goldberg, Purdom and Brown [18,26] show that for instances generated using the constant density model, the Davis-Putnam procedure runs in polynomial time. But the constant density model has been criticized [13] because it allows 0-literal clauses[1] with almost the same probability that it allows k literal clauses. Hence, the random instances generated according to $M_1(n, r, p)$ are either unsatisfiable with a high probability or satisfiable by a random assignment with a high probability. Franco [13] argues that the constant density model is therefore inferior to the constant component model.

For the purpose of analyzing the average case behaviour we can use the constant density model for generating the problem. However, for our purposes this generation mechanism does not suffice, as the intersection amongst the clauses is not guaranteed.

Another way of generating instances of NAESPI would be to use Bioch and Ibaraki's scheme [2] for generating all the self-dual functions, which in CNF representation would correspond to the NAESPI instances which are unsatisfiable. Satisfiable instances of NAESPI could then be obtained by removing a clause from the unsatisfiable instances.

[1] 0-literal clauses are the empty clauses.

An NAESPI instance is *symmetric* if all the variables belong to the same fraction of the total number of clauses and this property holds for every subset of clauses in the given instance. Symmetric instances are satisfiable in polynomial time by Khachiyan's algorithm and a variant of our basic algorithm (to be described in Section 5.1). A cursory look reveals that all the self-dual functions of seven variables are symmetric. Due to the lack of a rigorous study, at this juncture we cannot rule out the possibility that this is a not a good method of generating a test-bed of instances.

Next we propose a method of randomly generating NAESPI instances. To generate an instance of NAESPI with n clauses and m variables, we consider a clique of size n (with $\binom{n}{2}$ edges). Every variable belongs to an edge with probability p and does not belong to an edge with probability $1-p$. If L_v is the set of variables on the edges incident on node v, then clause v is defined to be L_v.

This generation scheme guarantees that the intersection property is obeyed, but it can generate instances in which there are clauses containing some other clause. As our algorithm is insensitive to the containment of clauses this seems like a viable method for generation. For any two clauses, the corresponding vertices in the graph share the variables that belong to the edge between the two vertices, thus guaranteeing intersection.

Our generation scheme can be visualized as a union of m edge labelled random graphs over the same set of vertices, such that each random graph has a label associated with all its edges (a label corresponds to a variable). The two most studied models of random graphs are $G(n,p)$ and $G(n,e)$, defined below.

Definition 12 (G(n,p)) *Given a positive integer n, $G(n,p)$ is the probability space on the graphs over n vertices with edge probability p where the events are mutually independent.*

In other words, if G_e is a graph over n vertices with exactly e edges then

$$P[G = G_e] = p^e(1-p)^{n-e}$$

Another widely studied model of random graphs is the class $G(n,e)$.

Definition 13 (G(n,e)) *$G(n,e)$ comprises all the graphs over n vertices containing exactly e edges where each graph is equally likely.*

From a theorem of Angluin and Valiant [4] it follows that the two models are practically interchangable when e is close to pn. Hence, we will consider only the $G(n,p)$ model. In the next section we describe a simple modification to the algorithm presented in Section 4. The modification is based on the observations of Fredman and Khachiyan [14].

5.1 A Variant

To recap, we begin by trying out all the 10^* possibilities at the top level. For every choice made we compute the sets P, N, PN. We denote the resulting subproblem by C_1. Next we compute all the unsafe variables in N (denoted U). Now we generate another subproblem C_2 obtained by collapsing all the variables in U. We solve C_2 recursively by trying out all the 10^* possibilities for clauses in N. We solve C_1 recursively but we only try those 10^* possibilities for which some variable in U is set to 0.

The first key observation is that we do not need to try 10^* possibilities for all the variables in a given stage. Let v be a variable which is unassigned in Stage *i*. In what follows we will change the definition of an unsafe variable slightly.

Definition 14 (Unsafe in P(N)) *A variable is called* unsafe in P(N) *if it belongs to more than* $\frac{n}{\epsilon}$ *clauses in P(N) for some* ϵ *respectively, where* n *is the total number of clauses in P and N.*

The following three cases are of interest to us and form the basis for the variant which we develop in this section.

i) If v is *unsafe* in both P and N then we just try setting v to 0 and 1. For each trial we get a subproblem of size $n(1-\frac{1}{\epsilon})$. Previously we were generating $O(n)$ subproblems when trying a value of 1 for the variable v. As the variable is either a 1 and 0 in the solution we move to Stage *i+1* after solving these two subproblems.
ii) If v is *unsafe* in P and *safe* in N then we first try setting v to 1, which results in a subproblem of size at most $n-1$. If we fail then v occurs as a 01^* in the solution. Here, we generate at most n subproblems each of size at most $\frac{n}{\epsilon}$. This follows from the fact that for each 01^* trial we have to consider only those clauses in N which contain the variable v. All the other clauses in N will already have a 1 assigned, because of the intersection property.
The case when v is unsafe in N and safe in P is symmetric.
iii) The case when v is safe in both P and N is treated in the same way as before.

The modification to our algorithm comprises of tackling these three cases separately as advocated by Fredman and Khachiyan. The correctness of the modified algorithm still follows from Theorem 3 and the bound on the running time follows from the next theorem.

Theorem 4 (Fredman & Khachiyan:96). *The algorithm terminates in* $n^{4o(\log n)+O(1)}$ *time.*

Proof. Here we will give only an outline of their proof. Fredman and Khachiyan choose $\epsilon = \log n$ thereby obtaining a bound of $n^{4o(\log n)+O(1)}$. They also argue that the third case never happens as there is always a variable which is unsafe either in P or in N. Otherwise all the clauses in P and N are of size greater than $\log n$. Hence, the running time for the algorithm is given by the following two recurrences:

– The first case is described by the following recurrence,

$$f(n) = 2f(n(1-\frac{1}{\epsilon}))+1$$

– This corresponds to the second case shown above. The complexity of this recurrence is dominated by the second term.

$$f(n) = f(n-1) + nf(\frac{n}{\epsilon})$$

They show that both these recurrences are bounded by $n^{\frac{4\log n}{\log\log n}+O(1)}$.

We conclude that by using the insight of Fredman and Khachiyan it is possible to improve the performance of the algorithm presented in Section 4 to achieve a running time of $n^{4o(\log n)+O(1)}$ by making the modifications described at the start of this section.

In the next section we show that Fredman and Khachiyan's algorithm plus the variant of our algorithm based on their approach terminates in

$$\max\{O(n^{3.87}), O(n^3\log^2 n), O(\frac{1}{p} - \frac{1}{\sqrt{p}})^{o(\log\frac{1}{p}-\frac{1}{\sqrt{p}})}\}$$

time for the class of random NAESPI.

6 Analysis

In this section we analyze the average case behaviour of the algorithm presented in the previous section. We critically use Lemmata 9, 10 and 11, whose proofs can be found in the appendix.

Theorem 5. *The algorithm presented in the previous section terminates in*

$$\max\{O(n^{3.87}, O(n^3\log^2 n), O(\frac{1}{p} - \frac{1}{\sqrt{p}})^{2+o(\log\frac{1}{p}-\frac{1}{\sqrt{p}})}\}$$

time on average, where p is the probability with which the initial input instance is generated.

Proof. Let $P_k(N_k)$ denote the number of clauses in sets $P(N)$ at Stage k. The following cases are of interest:

i) Before some stage i, $p \geq \max\{\frac{1}{P_i}, \frac{1}{N_i}\}$. By Lemma 11 and Lemma 9 there exists a variable which belongs to at least $P_i\frac{\sqrt{5}-1}{4}$ and $N_i\frac{\sqrt{5}-1}{4}$ clauses. Here, the first case of the algorithm is invoked, which generates two subproblems each of size at most $O(P)$ or $O(N)$. The number of subproblems is described by the following recurrence:

$$f(n) = 2f(n\frac{5-\sqrt{5}}{4})$$

which is $O(n^{1.87})$. As it takes $O(n^2)$ time to verify the solution for each subproblem, the total running time is $O(n^{3.87})$.

ii) At some stage $j \geq i$ either $p \leq \frac{\log P_j}{P_j^2}$ or $p \leq \frac{\log N_j}{N_j^2}$. By Lemma 10 and Lemma 9 there exists a variable which belongs to at most $\log n$ clauses in P_j. Therefore the number of subproblems of size $\frac{P_j}{\log P_j}$ generated in the second step of the algorithm is at most $\log n$ and the number of subproblems is given by the following recurrence:

$$f(n) = nf(\log n)$$

which is $O(n \log^2 n)$. As it takes $O(n^2)$ time to verify the solution for each subproblem, the total running time is $O(n^3 \log^2 n)$.

iii) Without loss of generality assume that $\frac{\log P_j}{P_j^2} \leq p \leq \frac{1}{P_i}$. The time taken in this case, by Theorem 4, is at most $O(P_i - P_j)^{o(\log P_i - P_j)}$. Given that $\frac{1}{P_j^2} \leq \frac{1}{P_i}$, we obtain $P_j \geq \sqrt{P_i}$. Substituting the lower bound on P_j in the previous equation, the time taken is at most $O(P_i - \sqrt{P_i})^{o(\log P_i - \sqrt{P_i})}$. Also, we have $p \leq \frac{1}{P_i}$, equivalently $P_i \leq \frac{1}{p}$. Hence, the number of subproblems in this case is at most $O(\frac{1}{p} - \frac{1}{\sqrt{p}})^{o(\log \frac{1}{p} - \frac{1}{\sqrt{p}})}$. As it takes $O(n^2)$ time to verify the solution for each subproblem, the total running time is $O(\frac{1}{p} - \frac{1}{\sqrt{p}})^{2+o(\log \frac{1}{p} - \frac{1}{\sqrt{p}})}$.

7 Conclusion

We presented a simple algorithm for determining the satisfiability of the NAESPI problem. We showed that the algorithm has a running time of $O(n^{2 \log n + 2})$. We also proposed a method for randomly generating NAESPI instances. We presented a variation of our basic algorithm for solving NAESPI based on the ideas of Fredman and Khachiyan [14]. We showed that the variant of our basic algorithm has the same complexity as the algorithm of Fredman and Khachiyan [14], i.e., $O(n^{4o(\log n)+O(1)})$. Furthermore, we analyzed the performance of our algorithms on the randomly generated instances of NAESPI and showed that the algorithm terminates in $\max\{O(n^{3.87}), O(n^3 \log^2 n), O(\frac{1}{p} - \frac{1}{\sqrt{p}})^{2+o(\log \frac{1}{p} - \frac{1}{\sqrt{p}})}\}$ time.

References

1. J. Bioch and T. Ibaraki. Complexity of identification and dualization of positive boolean functions. *Information and Computation*, 123(1):50–63, 1995.
2. J. Bioch and T. Ibaraki. Decomposition of positive self-dual functions. *Discrete Mathematics*, 140:23–46, 1995.
3. J. C. Bioch and T. Ibaraki. Generating and approximating nondominated coteries. *IEEE Transactions on parallel and distributed systems*, 6(9):905–913, 1995.
4. B. Bollobas. *Random Graphs.* Academic Press, 1985.
5. E. Boros, P.L. Hammer, T. Ibaraki, and K. Kawakami. Polynomial-time recognition of 2-monotonic positive boolean functions given by an oracle. *SIAM Journal on Computing*, 26(1):93–109, 1997.

6. E. Boros, V. Gurvich, L. Khachiyan, and K. Makino. On the complexity of generating maximal frequent and minimal infrequent sets. *In Proc. STACS*, LNCS 2285, 133–141, 2002.
7. J. deKleer and B. C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130, 1987.
8. C. Domingo, N. Mishra and L. Pitt. Efficient Read-Restricted Monotone CNF/DNF Dualization by Learning with Membership Queries. *Machine Learning*, 37(1):89–110, 1999.
9. T. Eiter and G. Gottlob. Identifying the minimum transversals of a hypergraph and related problems. *Siam Journal of Computing*, 24(6):1278–1304, 1995.
10. T. Eiter and G. Gottlob. Hypergraph Transversal Computation and Related Problems in Logic and AI. *In proceeding of Logics in Artificial Intelligence, JELIA*, LNCS 2424:549–564, 2002.
11. T. Eiter and K. Makino. On computing all abductive explanations. *In Proceeding 18th National Conference on Artificial Intelligence, AAAI-02*, 62–67, 2002.
12. W. Feller. *Introduction to Probability Theory and its Applications*. John Wiley and Sons, 3rd edition, 1967.
13. J. Franco. On the probabilistic performance of the algorithms for the satisfiability problem. *Information Processing Letters*, pages 103–106, 1986.
14. Michael L. Fredman and Leonid Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *Journal of Algorithms*, 21(3):618–628, Nov. 1996.
15. H. Garcia-Molina and D. Barbara. How to assign votes in a distributed system. *Journal of the ACM*, 32:841–860, 1985.
16. D. Gaur and R. Krishnamurti. Self-duality of bounded monotone boolean functions and related problems. In *The Eleventh Conference on Algorithmic Learning Theory*, Lecture Notes in Computer Science (subseries LNAI), pages 209–223, 2000.
17. D. Gunopulos, R. Khardon, H. Mannila, H. Toivonen. Data mining, Hypergraph Transversals, and Machine Learning. *In Proc. Symposium on Principles of Database Systems*, pages 209–216, 1997.
18. A. Goldberg, P. Purdom, and C. Brown. Average time analysis for simplified davis-putnam procedures. *Information Processing Letters*, 15(2): pages 72–75, 1982.
19. V. Gurvich and L. Khachiyan. Generating the irredundant conjunctive and disjunctive normal forms of monotone boolean functions. Technical Report LCSR-TR-251, Dept. of Computer Science, Rutgers Univ., Aug. 1995.
20. T. Ibaraki and T. Kameda. A boolean theory of coteries. *IEEE Transactions on Parallel and Distributed Systems*, pages 779–794, 1993.
21. T. Eiter, G. Gottlob, and K. Makino. New results on monotone dualization and generating hypergraph transversals. *Proceedings of the* 34th *Annual ACM Symposium on Theory of Computing*, pages 14–22, 2002.
22. K. Makino. *Studies on Positive and Horn Boolean Functions with Applications to Data Analysis.* PhD thesis, Kyoto University, March 1997.
23. K. Makino. Efficient dualization of $O(\log n)$ term disjunctive normal forms. *Discrete Applied Mathematics*, 126(2-3), pages 305-312, (2003).
24. K. Makino and T. Ibaraki. The maximum latency and identification of positive boolean functions. In D. Z. Du and X. S. Zhang, editors, *ISAAC 1994, Algorithms and Computation*, volume 834 of *Springer Lecture Notes in Computer Science*, pages 324–332.
25. H. Mannila and K. J. Räihä. An application of armstrong relations. *Journal of Computer and System Science*, 22:126–141, 1986.

26. P. W. Purdom and C. A. Brown. The pure literal rule and polynomial average time. *SIAM Journal on Computing*, 14:943–953, 1985.
27. R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.
28. P. D. Wendt, E. J. Coyle, and N. C. Gallagher Jr. Stack filters. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34(4):898–911, 1986.

Appendix

Let V_i be the indicator variable whose value determines if the i^{th} vertex has been covered by some edge in a randomly generated graph G. E_j is the indicator variable which determines if the j^{th} edge has been chosen.

$$|V| = \sum_{i=1}^{n} V_i$$

and

$$|E| = \sum_{j=1}^{\binom{n}{2}} E_j$$

Lemma 7. $Pr[|V| \leq k] \geq Pr[|E| \leq \frac{k}{2}]$.

Proof. The claim follows from the fact that if $\frac{k}{2}$ edges are present then at most k vertices are covered.

Let $G = (n, e)$ be a random graph over n vertices with exactly e edges where $e \leq n$. If all the edges in G are equally likely then the average number of vertices covered in G is $O(e)$.

Lemma 8. *If the number of vertices covered in G is at most $n\frac{\sqrt{5}-1}{2}$ then the expected number of vertices covered satisfies $\mathcal{E}[V] \geq \frac{\sqrt{5}-1}{2}e$.*

Proof. If x is the fraction of vertices covered by some $e' < e$ edges then the probability that we increase the number of vertices covered for each edge chosen is at least $1 - x^2$. $|V| \leq n\frac{\sqrt{5}-1}{2}$ implies that $x \leq \frac{\sqrt{5}-1}{2}$, and therefore the probability of success is at least $1 - (\frac{\sqrt{5}-1}{2})^2$. This implies that the expected number of vertices covered is at least $\frac{\sqrt{5}-1}{2}e$.

Next we use the Chebyshev inequality to bound the tail probability of Lemma 8. Without loss of generality we assume that the probability with which we increase the number of vertices covered is $\frac{\sqrt{5}-1}{2}$. Our experiment can also be visualized as flipping a coin e times where the probability of head occurring (denoted p) in each individual trial is $\frac{\sqrt{5}-1}{2}$. This is the binomial distribution with mean $\mu_x = ep$ and standard deviation $\sigma_x = \sqrt{epq}$. For the next lemma we assume that $p = \frac{\sqrt{5}-1}{2}$ and $q = 1 - p$.

Lemma 9. *If e edges are covered then with probability $1-\frac{4q}{ep}$ at least $\frac{ep}{2}$ vertices are covered.*

Proof. The Chebyshev inequality asserts that,

$$Pr[|X-\mu_x| > t\sigma_x] \leq \frac{1}{t^2}$$

choosing $t\sigma_x = \frac{ep}{2}$, we get $\frac{1}{t^2} = \frac{4q}{ep}$. Hence, with probability $1-\frac{4q}{ep}$ we cover at least $ep - \frac{ep}{2} = \frac{ep}{2}$ vertices. Observe that this probability goes to 1 as $e \rightarrow \infty$.

The probability that k edges are choosen (for a given label) is given by the binomial distribution,

$$Pr[E=k] = \binom{m}{k} p^k (1-p)^{m-k}$$

where $m = \binom{n}{2}$. Next, we use the Demoivre-Laplace theorem [12] to compute the probabilities we are interested in.

Theorem 6 (DeMoivre-Laplace). *Suppose $0 < p < 1$ depends on n in such a way that $npq \rightarrow \infty$ as $n \rightarrow \infty$. If $0 < h = x\sqrt{pqn} = o(pqn)^{\frac{2}{3}}$ and if $x \rightarrow \infty$, then $Pr[|X-np| > h] \sim \frac{e^{\frac{-x^2}{2}}}{x\sqrt{2\pi}}$.*

The DeMoivre-Laplace theorem is obtained by approximating the binomial distribution with the normal distribution. For a random variable X it gives the probability that $X - np \geq x$, where x is measured in steps of $\sqrt{npq}$, where np is mean and $\sqrt{npq}$ is the standard deviation of the binomial distribution.

Lemma 10. *Let $m = \binom{n}{2}$. For $p \leq \frac{\log n}{m}$, $Pr[E \leq 2\log n] \geq 1 - \frac{1}{\sqrt{2\pi \log n} e^{(\frac{\log n}{2})}}$.*

Proof. Let $p = \frac{\log n}{m}$ then $\mu_x = mp = \log n$ and $\sigma_x^2 = mpq = \log n(1-\frac{\log n}{m}) \sim \log n$ for large m. Choosing $h = \log n$, we get $x = \frac{\log n}{\sqrt{\log n}} = \sqrt{\log n}$.

Then by the DeMoivre-Laplace theorem,

$$Pr[E \geq 2\log n]_p \sim \frac{1}{\sqrt{2\pi \log n} e^{(\frac{\log n}{2})}}$$

and for $p' < p$

$$Pr[E \leq \log n]_{p'} \geq Pr[E \leq \log n]_p$$

Hence, the result.

Similarly,

Lemma 11. *Let $m = \binom{n}{2}$. For $p \geq \frac{1}{n}$, $Pr[E \geq \frac{n}{4}] \geq 1 - \frac{4}{\sqrt{2\pi n} e^{(\frac{n}{32})}}$.*

Proof. Let $p = \frac{1}{n}$ then $\mu_x = mp = \frac{n-1}{2}$ and $\sigma_x^2 = mpq = n(1 - \frac{1}{n}) \sim n$ for large n. Choosing $h = \frac{n}{4}$, we get $x = \frac{n}{4\sqrt{n}} = \frac{\sqrt{n}}{4}$.

Then by the DeMoivre-Laplace theorem,

$$Pr[E \leq \frac{n}{4}]_p \leq \frac{4}{\sqrt{2\pi n} e^{\frac{(\frac{\sqrt{n}}{4})^2}{2}}}$$

and for $p' > p$

$$Pr[E \geq \frac{n}{4}]_{p'} \geq Pr[E \geq \frac{n}{4}]_p$$

Hence, the result.

Detecting Deception in Intelligent Systems I: Activation of Deception Detection Tactics

Gregory Johnson Jr. and Eugene Santos Jr.

University of Connecticut
Booth Engineering Center for Advanced Technology (BECAT)
371 Fairfield Road, Unit 2031
Storrs, CT 06269-2031
{gjohnson,eugene}@cse.uconn.edu

Abstract. The continued research, development and acceptance of intelligent systems for diagnosis and decision support has uncovered many practical considerations for the interaction of intelligent, autonomous agents. One such consideration is the possibility of an agent intentionally transmitting misinformation to other agents or to a human decision maker to achieve its own goals. Most intelligent, knowledge-based systems to date have assumed all experts involved in system development operate toward a shared goal and operate in good faith. We wish to challenge these assumptions and consider cases where agents in the system are not assumed to operate in good faith and therefore misinformation is a possibility. Most literature devoted to deception in agent systems focuses on building trust relationships through a history of interaction. We apply models of deception and deception detection from psychology and cognitive science to intelligent multi-agent systems. We focus on a model for deception detection in which the detection and analysis of deception are decoupled. Finally, we introduce a novel method for the detection of deception in multi-agent systems based on correlations between the agents in the system.

1 Introduction

With the increased acceptance of knowledge-based intelligent systems for diagnosis and decision support, the dependence of human decision makers upon such systems has increased as well. As we become more dependent upon these systems the impact of deceptive information offered by these systems becomes impossible to ignore. Particularly with the interaction of multiple intelligent agents in multi-agent systems, it becomes increasingly difficult to validate information offered by these systems. This creates a scenario in which incorrect information can be introduced into the system causing the knowledge-based system to reach incorrect conclusions in order to mislead the decision maker. Historically, this has been ignored by the developers of expert systems by assuming that all contributing experts operate in good faith toward a common goal. Thus, there arises a need to monitor these knowledge-based systems for deceptive information. We briefly identify a theory of deception and types of deception based on work by Bell and Whaley [12,3,2], this is perhaps the only work to date in forming a general theory of deception. We further identify a model of deception detection, based on work by Johnson et. al. [8], which is applicable to intelligent systems which not only attempts to detect deception but identify the deception tactic as well as the goal of the deception. The first step in this model of deception detection, as will be elaborated upon later, is named Activation and is concerned with the detection of unexpected results or some piece of information which is suspect. In this paper, we are specifically interested in identifying methods for implementing this first step in intelligent systems without regard to the particular tactics or goals of the deception. Specifically, we employ an algorithm from the collaborative filtering domain to predict agent opinions from their correlation with other agents in the past. Finally, we evaluate two methods of detecting unexpected results in a traditional expert system with one expert knowledge base. The first method consists of modelling the knowledge base directly through data mining techniques on stored responses from that knowledge-base. The second technique uses the same algorithm from the collaborative filtering domain to predict the response of the knowledge-based system based on the correlation of the system and the models in the past.

The general design of systems which we intend to apply the deception detection algorithms consists of an expert knowledge base over a specific set of random variables. The expert constructing the knowledge base may introduce intermediate random variables during the construction of the knowledge-base, however, a set of evidence and a set of hypothesis random variables are fixed at system design time and are not changed. The expert knowledge-base is given a set of observations or instantiations of some subset of the evidence random variables. The expert knowledge-base then produces a posterior probability distribution over the hypothesis random variables and possibly the evidence random variables which were not observed. This expert knowledge-base may be inside a classical expert system or within an intelligent agent.

The empirical results obtained from our experiments indicate that it is indeed possible to detect deceptive results produced by an agent in a multi-agent system with reasonable success. These results also indicate that, in the classic expert system, modelling the intelligent system with data mining techniques is the most accurate method of predicting responses. Finally, we note that the prediction of agent opinions is not only useful in identifying deceptive disinformation, but has the added benefit in identifying unintentional misinformation due to an incomplete or invalid knowledge base.

2 Deception

Detecting deception is a difficult task. In fact, humans can detect when someone is being deceptive only a little better than fifty percent of the time (45-65%)[7]. Research suggests that when people use electronic media to interact, the deceived is even less effective at detecting when another is trying to deceive him [7].

In this section we present an overview of a theory of deception which has been developed by researchers in cognitive science and psychology. These theories appear to hold promise for implementation in knowledge-based intelligent systems.

2.1 Theory of Deception

Deception is defined by Burgoon and Buller to be a "deliberate act perpetrated by a sender to engender in a receiver beliefs contrary to what the sender believes is true to put the receiver at a disadvantage" [5]. This definition, and most other accepted definitions of deception, clearly include the intent of the deceiver to achieve some goal as a requirement for deception. The conveyance of misleading information is not sufficient to conclude deception. Therefore, we adopt an intentional stance [6] when attempting to detect deception. That is, we presume that all agents have particular beliefs about the environment and perform actions (e.g. deception) to achieve one or more goals. We unconsciously use this simple yet powerful presumption everyday. Intentionality allows us to determine the goals of others and therefore predict their behavior. In fact, to succeed, deceivers must also rely upon intentionality when crafting their deceptions.

2.2 Types of Deception

Bell and Whaley [3,12,2] identify two broad categories of deception, dissimulative and simulative. Dissimulative deceptions attempt to 'hide the real' whereas simulative deceptions attempt to 'show the false.' Each may be further divided into three categories and each category is referred to as a deception tactic (also see [9] for an expanded presentation of the six deception tactics with respect to intelligent systems). The attributes in the environment on which the deception is focused is referred to as the 'deception core' [8].

Dissimulation is characterized by an attempt to 'hide the real' and deceptions of this type fall into three categories[3,12,2]:

- **Masking** attempts to make a feature of the environment invisible either by blending with the background or avoiding detection.
- **Repackaging** attempts to hide the real by making something appear to be what it is not. This may be achieved by making something appear dangerous, harmless or irrelevant.
- **Dazzling** is employed when the feature in the environment is known to exist. Specific examples are when a military commander sends multiple movement orders for troops via radio, only one of them being the real orders, knowing the enemy will hear all of the messages. In intelligent systems, this may be encountered when a diagnosis includes too many faults for a human decision maker to respond.

Simulation is characterized by an attempt to 'show the false' and may be used in conjunction with hiding. Three types of simulative deception include [3,12,2]:

- **Mimicking** is achieved when something is made to appear to be what it is not. This can be employed not only to hide but also to achieve an advantage. This tactic can be used to lure prey or elicit additional actions in which the target unwittingly assists the deceiver achieve a goal. In intelligent systems this tactic may be employed to hide the source of a fault but not the fault itself.
- **Inventing** a new reality is another tactic for showing the false as in the case of the wolf in sheep's clothing. Inventing in intelligent systems is difficult to achieve since introducing new random variables into the system is easy to detect with some constraints in the design of the system.
- **Decoying** is another simulative tactic which is aimed at luring the target of the deception away from discovering the real. For example, some birds will lure predators away from their nest of babies. One phrase which describes this tactic is 'creating a diversion.' This tactic is often used when the perpetrator can leave several courses of action open, one of which to be chosen at the last moment.

2.3 Detecting Deception

Paul Johnson et al. [8] identify several strategies for detecting deception generalizing from work by Mawby and Mitchell [10]. These fall into two broad groups: strategies which are based on detecting the process of deception and strategies based on analyzing information in the altered environment. For intelligent systems, we focus on strategies for detecting deception which are concerned with detecting inconsistencies in the altered environment. The following paragraphs introduce the detection strategies most relevant to detecting deception in intelligent systems:

- **Preemption Based.** This strategy seeks to limit the amount of deception that may be practiced by an agent by limiting the degrees of freedom an agent has in manipulating the environment. This includes limiting any changes to the knowledge-base once it has passed verification and validation.
- **Intentionality Based.** This deception detection strategy acknowledges that agents have beliefs and knowledge of their environment as well as goals. Deceptions practiced are done so to achieve one or more of those goals. This strategy focuses on an agent's goals and the actions that agent can take. The knowledge of an agent's beliefs and goals are also powerful in predicting the actions that agent will take.

2.4 A Model for Detecting Deception

Johnson and Grazioli [8] propose the following model for detecting deception. It consists of four domain independent steps:

1. **Activation.** The target of the deception observes an attribute in the environment which is not what was expected.

2. **Detection.** The target of the deception employs detection strategies to produce hypotheses about suspected manipulations in the environment.
3. **Editing.** The target of the deception edits the representation of the observed environment to one which is consistent with the hypothesized manipulations.
4. **Reevaluation.** The target of the manipulation chooses an action based on the revised knowledge of the environment.

The focus of this paper is the implementation of the Activation step within a multi-agent system. Further research will focus on identifying the deception tactic and using the detected deceptions to build or enhance a model of the deceptive adversary agent for use in predicting future agent actions.

3 Detecting Deception in Intelligent Systems

We would like to apply the deception detection tactics and models to the Multi-Agent System framework. Specifically, we are interested in methods for detecting intentional deceptions aimed at causing the system to fail. The following sections outline a possible method for implementing the Activation phase in the model of fraud detection developed by Johnson and Grazioli [8]. In the case of stand-alone knowledge-based systems, we can use data mining algorithms to build models of the knowledge-base for predicting future responses. The availability of the opinions of other agents in multi-agent systems allows deception to be detected through the comparison of agent opinions when those agents have shared knowledge over the same domain. In fielded systems, one would expect a high degree of correlation between agents with similar knowledge. Therefore, the comparison of agent posterior distributions, given the same observations, appears to be a promising method for the detection of deceptive information. Furthermore, this approach is independent of the chosen knowledge representation or aggregation scheme. It may be applied to any probabilistic knowledge representation.

3.1 Activation

In order to detect deceptive information in multi-agent systems, one must have some expectations for a particular agent's opinion given some evidence. That is, we need some way of predicting what an agent will offer as an opinion. To achieve this goal, we employ techniques developed from research in collaborative filtering systems. Collaborative filtering techniques are most often used in recommendation systems. These recommendation systems use the past behavior or opinions of other users of the system with similar tastes or behaviors to recommend items for a particular user. For example, consider a movie recommendation system. A group of users of a movie recommendation system rate their opinion of movies which they have viewed in the past on a five point scale. The movie recommendation system uses these ratings to find correlations between users who have seen some of the same movies in the past. From these correlations, the system attempts to select movies a particular user has not viewed (or rated) and is likely to enjoy. Likewise, in multi-agent systems, we intend to predict the opinions of expert agents in the system from their past opinions and the opinions of other agents in the system.

Preemption Based Strategy
Attempts at deception by inventing or masking tactics can be detected by a Preemption based strategy. The architecture of the multi-agent system allows for a design decision between two alternatives which limit the freedom of an agent to *invent* new features in the environment or *repackage* features which the agent would like to keep hidden. First, the system designer may require that all agents offer an opinion for all hypotheses. In this case, concealment is detected simply by ensuring each agent offers a probability distribution that includes all hypothesis random variables. Alternately, if the problem domain is sufficiently large, some agents may not have the required knowledge to evaluate all hypotheses included in the system. The next section introduces a method for comparing agent opinions which requires us to store the entire history of agent posterior distributions. With this information, deception may be suspected when the posterior distribution offered by an agent does not contain a hypothesis (or hypotheses) included in previous posterior distributions produced by that agent. In both situations an agent is prevented from inventing a feature in the environment since all random variables in the system are agreed upon at system design. If an agent includes an unknown random variable (or one for which it has never offered an opinion before) in their posterior distribution, this is easily detected as *inventing*.

Intentionality Based Strategy
The main task in the Activation phase of deception detection is to monitor the 'environment' (in this case in the form of agent posterior probability distributions) for deviations from what is expected. This implies one has some expectation as to what an agent opinion 'should' be. That is, we need to be able to predict agent opinions, at least reasonably accurately. Just as it seems reasonable, in the context of collaborative filtering, that a good way to find interesting content is to find people with similar interests, it seems reasonable to assume we may anticipate an agent's opinion from the opinions of agents with similar knowledge. With this assumption, we may use the opinions of several agents to validate each other. In the case of the stand-alone knowledge-based system, models learned from past responses of the system can be used to predict agent opinions. An elegant solution is obtained through the use of the Pearson correlation coefficient and the GroupLens prediction equation [11] which is used to predict the utility of a news item from the opinions of other users in a newsgroup as shown in Equation (1). This metric can be adopted in a multi agent system to attempt to predict what an agent will offer for it's opinion.

Comparing Agent Opinions
The Pearson correlation coefficient (Equation (1)) can be applied in the multi-agent framework to obtain a measure of the degree of correlation between two agents. We can consider each possible hypothesis individually. For hypothesis $h =$

true we store a probability value from each agent for each set of instantiations of evidence random variables (r.v.s). So A_i corresponds to the posterior probability of $h = true$ offered agent A for the i^{th} set of instantiations of evidence r.v.s. Likewise, B_i is the probability agent B assigns to the event $h = true$ for the same instantiation of r.v.s. In the same fashion we consider every possible state for all hypotheses for which the agents offer opinions. The equation returns a weight between -1 and 1. If the two agents agree completely, their correlation coefficient would be 1, if they never agree, -1. Also, $\bar{A}$ is the average of all probabilities A has assigned to a hypothesis over all i sets of instantiations of evidence r.v.s and likewise for $\bar{B}$. In the case where an agent does not agree with the majority of other agents, we can use this metric to determine the average correlation of the disagreeing agent from the agreeing agents.

$$\begin{aligned} r_{AB} &= \frac{Cov(A,B)}{\sigma_A \sigma_B} \\ &= \frac{\sum(A_i - \bar{A})(B_i - \bar{B})}{\sqrt{\sum_i (A_i - \bar{A})^2}\sqrt{\sum_i (B_i - \bar{B})^2}} \end{aligned} \tag{1}$$

Expected Agent Opinions

The Pearson correlation coefficient is then used in the GroupLens prediction equation [11] shown in Equation (2). Given a sufficient record of posterior distributions over a set of hypotheses returned by all agents and the opinions of all the other agents, we can produce an estimate of what we expect a particular agent should have returned for each hypothesis. If the agent offers a probability for that hypothesis which is extremely different than the expected probability, the next step in deception detection is carried out to search for patterns in the agent's posterior distribution which match one of the deception tactics. Resnick et al. note that this prediction equation is robust with respect to certain interpretations of the rating scale for the news items. For instance, if one user consistently rates news items in range 1-3 and another rates items in the range 3-5 and the two users would be perfectly correlated otherwise, a score of 5 for the second user would predict a score of 3 for the first user. Likewise for agent opinions, the prediction equation is invariant under shifting and scaling of agent opinions. If one agent consistently assigns probabilities in the range of 0.4 - 0.7 for a hypothesis and another agent assigns probabilities in the range 0.5 - 0.8 and the agents are perfectly correlated otherwise, a probability of 0.8 assigned by the second agent would result in a prediction of 0.7 for the first agent.

$$A_{X_{predicted}} = \bar{A} + \frac{\sum(B_{i_X} - \bar{B}_i) r_{AB_i}}{\sum_i |r_{AB_i}|} \tag{2}$$

Model-Based Approach

In the collaborative filtering domain, probabilistic methods attempt to predict the expected value of a vote given what is known about the user [4]. Likewise, a probabilistic model-based approach can be used to predict agent opinions for the purposes of deception detection. Given a history of an agent's prior activity in a domain, a model of that agent can be constructed using data mining algorithms and this model can be used to predict future responses from that agent. This approach is particularly useful in the single-agent case where there are no other agents with which to compare responses. The models obtained through these data-mining learning algorithms may be incomplete however, and may not model the agent perfectly. Therefore, we should only view the models as a guide to detecting deception among agents and not as a replacement for agent knowledge-bases.

4 Empirical Evaluation

An empirical evaluation of the intended implementation was conducted to implement and test the activation algorithms to determine if it is possible to detect deception in intelligent systems with our current model. First, experiments were conducted in a multi-agent environment using the GroupLens[11] prediction equation. Next, experiments were conducted which attempted to use collaborative filtering algorithms to predict the opinions of a single agent. In the single agent case, two methods of predicting agent opinions were tested. The two predictors included in our experiments were a single Bayesian network model of the agent knowledge-base and the output of two models of the agent knowledge-base combined by using the GroupLens prediction algorithm. Studies in the field of collaborative filtering suggest the Bayesian model is a better predictor [4]. The empirical evaluation was conducted using the Alarm [1] Bayesian network. The Alarm network contains 37 random variables (nodes) and 46 arcs. The Alarm network is clearly divided into 8 diagnoses, 16 findings and 13 intermediate random variables. This network was selected for it's moderate size and clear structure.

4.1 Assumptions

We assume that deceptions occur with a low base-rate. That is, the deceiver acknowledges that to be successful it must be trusted and constant deception is perceived as unreliability. Therefore, the deceiver most often offers good-faith opinions and only offers deceptive opinions only when it is necessary to achieve one or more goals. We assume the opinions given by an agent during system validation are not deceptive and therefore it is reasonable to assume we can collect a substantial number of honest opinions with which we can determine the correlation between multiple agents or build a model of a single agent.

With respect to the distribution of error between our model and the true agent opinion, we assume the error is normally distributed. We make this assumption to take advantage of a property of normal distributions with respect to the standard deviation of the distribution. This property states that approximately, 68.3% percent of the data samples are within one standard deviation of the mean of a normal distribution. Likewise, approximately 95.4% and 99.7% of samples are within two and three times the standard deviation of a normal distribution, respectively. These properties hold for all normal distributions regardless of their particular variance or standard deviation. Therefore, we hypothesize that if the difference between an opinion offered by an agent and the predicted value is greater than three times the standard deviation then it is reasonable to activate the next step in deception detection. That is, if the prediction error is greater than three times the standard deviation of error for that random variable then it is quite possible the offered opinion is deceptive. The next step in our adopted model of deception detection attempts to classify the detected deception(s) into the six deception tactics. If a suspected deception does not match any of the deception tactics it is presumed to be a false positive detection of deception. For the purposes of this paper, however, we focus on Activation, the first step of the deception detection model.

4.2 Multi-agent Detection

In the multi-agent case, we make the additional assumption that agent opinions will be correlated, at least to some degree. This assumption seems reasonable when we consider that agents in a particular domain are often educated or trained in similar institutions. For example, one would expect a group of medical doctors trained at different medical schools to agree on diagnoses for the vast majority of cases.

Experimental Setup

Multiple agents were simulated by introducing perturbations into the conditional probability tables of the Alarm[1] network. Note, the structure of the Bayesian networks in these agents is not changed and therefore is identical for all agents. Random (selected from a uniform distribution) real numbers from the interval $[-0.1, 0.1]$ (and then a second experiment was conducted with real numbers from the interval $[-0.2, 0.2]$) were added to each conditional probability of each table. The conditional probability tables were then normalized to ensure each represented a valid probability distribution. Ten agents were generated in this fashion. Next, two sets of 1000 observations were generated by randomly selecting a random number (1-10) of random variables and instantiating each with a randomly selected state. All random selections were generated from the uniform distribution. One set of observations serves as a training set to compute the Pearson correlation coefficient and the standard deviation of the prediction error while the other set serves as a test set to evaluate the algorithm's ability to accurately detect deceptions. Ten simulated agents were generated for this experiment. The Pearson correlation coefficient between the Alarm network and each simulated agent was computed for each state of each random variable using the opinions obtained using the random observations in the training set.

Next, for each of the observations in the test set, the opinions of each of the ten simulated agents were used to predict the opinion obtained from the original Alarm network. A simple deception distribution was simulated by perturbing the posterior probability distribution computed by the Alarm network. That is, for each random variable, the posterior probability for the first state was reported to be that of the second, and so on. If the error in predicting the Alarm network response was greater than three times the standard deviation of the prediction error for that random variable state, this was considered a false activation. Likewise, if the prediction error of the simulated deception distribution was greater than three times the standard deviation for the prediction error for that random variable state, this was considered successful detection of deception.

Multi-agent Results
Tables 1 and 2 show a summary of the results obtained from our experiments with multi-agent deception detection. Due to the volume of data collected, only this summary is shown here. These tables show the minimum, maximum, average and median absolute value of correlation coefficients between the ten simulated agents and the Alarm network (for all 105 random variables states in the Alarm network) in each of the two experiments. Also in these tables are the minimum, maximum, average and mean values for the standard deviation of prediction error for each experiment. A multiple of this standard deviation is used as a threshold to detect deceptions. The last two rows in these tables show the minimum, maximum, average and mean fraction of deceptions detected and false activations reported.

These results show that there is indeed a high correlation between the simulated agents produced for this experiment. Furthermore, and more importantly, these results show that if there is a high degree of correlation between agents, as we have assumed, it is possible to accurately detect deceptions by individual agents using the GroupLens prediction algorithm. The results for the first multi-agent experiment show the average standard deviation of error for predictions to be approximately 0.07. On average, in this case, we can detect manipulations in the environment as small as 0.22. Noting that all the states of each random variable are inextricably bound to each other, Tables 3 and 4 show a summary of results for the state of each random variable which the algorithm was most successful at detecting deceptions. Clearly, if deception is suspected for one state of a random variable, the posterior probability of all other states of that random variable become suspect. Therefore, one could argue that we need only detect deception in the opinion for one state of a random variable in order to successfully detect a deceptive opinion. In the first experiment, our approach was able to detect an average of 68% of the simulated deceptions. In the second experiment, an average of 59% of the simulated deceptions were detected. In both experiments false activations were reported in approximately 2% of the cases on average. The worst case for false activations in both experiments was approximately 6.5%. The GroupLens prediction algorithm performed well in predicting agent opinions in our experiments. This allowed our deception detection algorithm to detect a respectable number of the simulated deceptions with a reasonable number of false activations.

Table 1. Summary table of results for agents generated with perturbations in the interval $[-0.1, 0.1]$.

	Min	Max	Mean	Median
Coefficients	0.389980102	0.998693299	0.911652591	0.93938709
Prediction error σ	0.003411243	0.152786685	0.074216888	0.070682193
Deceptions	0.110352673	1	0.683922583	0.708381171
False Activations	0	0.066746126	0.022965879	0.019953052

Table 2. Summary table of results for agents generated with perturbations in the interval $[-0.2, 0.2]$.

	Min	Max	Mean	Median
Coefficients	0.040879821	0.996061714	0.849596795	0.884381314
Prediction error σ	0.00662076	0.248520976	0.112183338	0.10610505
Deceptions	0.066210046	1	0.592662416	0.575147929
False Activations	0	0.067207416	0.017989745	0.017045455

Table 3. Summary table of results for agents generated with perturbations in the interval $[-0.1, 0.1]$. These results include only the most accurately detected state of each random variable.

	Min	Max	Mean	Median
Deceptions	0.309859155	1	0.825790735	0.859931114
False Activations	0.001133787	0.061484919	0.025495806	0.020954598

Table 4. Summary table of results for agents generated with perturbations in the interval $[-0.2, 0.2]$. These results include only the most accurately detected state of each random variable.

	Min	Max	Mean	Median
Deceptions	0.252293578	1	0.768870872	0.802497162
False Activations	0	0.067207416	0.020321088	0.020713464

Finally, we adjust the threshold for detection of deceptions for each random variable state to be four times the standard deviation of error for predictions for that r.v. state. Tables 5 and 6 show the results obtained by making this adjustment. These results show this adjustment in threshold did significantly reduce the number of false activations. However, this reduction in false activations came at a cost of a reduced number of detected deceptions.

Table 5. Summary table of results for the first experiment using four times the prediction error standard deviation as a threshold.

	Min	Max	Mean	Median
Deceptions	0.09556314	1	0.643633997	0.633918129
False Activations	0	0.025433526	0.005645427	0.004581901

Table 6. Summary table of results for the second experiment using four times the prediction error standard deviation as a threshold.

	Min	Max	Mean	Median
Deceptions	0	1	0.520044526	0.46627907
False Activations	0	0.027713626	0.004233501	0.001149425

4.3 Single Expert

To construct models of the Alarm network, thirty thousand records were generated from the Alarm network using a random number (of between two and ten) of randomly selected observed random variables. Observed random variables were selected from the nodes designated as measurement nodes (or findings)[1]. Models of the Alarm network were obtained from this data using the Bayesian Networks Tools in Java [1] implementation of the $K2$ algorithm and the BN Powersoft [2] implementation of the CBL_2 learning algorithm. Note, both models were obtained from the same data. These models were learned with no particular ordering of random variables. That is, no domain knowledge was used to obtain the models of the Alarm network. Next, the models and prediction algorithm were tested against the results from the Alarm network using one thousand test cases generated at random and separately from the data from which the networks were constructed. In this case, observed random variables were selected randomly from the entire network. Note, some random variable states were not included in the CBL_2 model. This may be corrected by adjusting the threshold supplied to the learning algorithm to be more sensitive to weak connections in the network. For the purposes of these experiments, the default values were used for all data-mining algorithms.

The results of the models from the test set were then used to compute the correlation coefficient for the GroupLens prediction algorithm. Next the GroupLens algorithm was used to predict the Alarm network results for each of the sets of observations in the test set.

Evaluating the Models

The results obtained from the $K2$ and CBL_2 models were evaluated by computing the Mean Error, Error Standard Deviation and Mean Square Error of each model with the results obtained from the Alarm network providing the oracular

[1] Bayesian Network Tools in Java was written by the Kansas State University KDD Lab and is available at `http://bndev.sourceforge.net/`.

[2] Bayesian Network PowerSoft was written by Jie Cheng and is available at `http://www.cs.ualberta.ca/~jcheng/bnsoft.html`.

distribution. The results of the prediction equation were used to combine the results from each of the models and were likewise compared to those obtained from the Alarm network. The model with the smallest standard deviation of error holds the greatest potential for detecting deception. Due to the volume of data obtained, only a small representative portion is shown here (see Tables 7, 8 and 9).

These results suggest that even though the Mean Squared Error (MSE) for both the $K2$ and GroupLens prediction are nearly comparable, the standard deviation for the $K2$ model is generally less than that of the GroupLens algorithm. This indicates that, overall, the error for GroupLens is distributed more widely than that of the $K2$ model. These results also suggest that the CBL_2 model generally produced a smaller MSE and a smaller standard deviation than both the $K2$ and GroupLens predictors. This indicates the CBL_2 model is likely to be the most accurate predictor of agent opinions.

Table 7. The minimum, maximum, mean and median values for the mean error, error variance and error standard deviation for all 105 random variable states of the K2 model.

	Mean	Median	Minimum	Maximum
Mean Error	0.001314032	0.008954107	-0.702853112	0.778320431
Error σ	0.26074353	0.266048725	0.024737753	0.439759621
Mean Squared Error	0.12968684	0.102540765	0.001055072	0.708386411

Table 8. The minimum, maximum, mean and median values for the mean error, error variance and error standard deviation for 99 random variable states of the CBL model.

	Mean	Median	Minimum	Maximum
Mean Error	-0.000229167	-0.014736709	-0.484510498	0.496652082
Error σ	0.207239234	0.194327669	0.054117194	0.408614851
Mean Squared Error	0.080397612	0.062100575	0.003001032	0.394648389

Table 9. The minimum, maximum, mean and median values for the mean error, error variance and error standard deviation for the 99 random variable states predicted by the GroupLens prediction algorithm.

	Mean	Median	Minimum	Maximum
Mean Error	-0.005583937	-0.000504864	-0.250705326	0.05557251
Error σ	0.389925912	0.323387973	0.031510618	1.396537023
Mean Squared Error	0.118060338	0.109149876	0.003776583	0.265790029

Activation Test

Next, a copy of the computed opinions from the Alarm network was stored in a database and the opinions for the StrokeVolume variable were shifted to simulate a deception at that node. That is, the response for the first state was reported to be that of the second and so on. This node was selected since the standard deviation of error in predicting StrokeVolume was one of the best at 0.05 for state *high*. Each opinion was considered deceptive if the offered probability differed from the predicted probability by more than three times the computed standard deviation of error for that random variable state. The CBL_2 model performed the best at detecting deceptions. For CBL_2, deception was detected with a success rate of approximately 78% for state *high*, 67% for state *normal* and 33% for state *low*. However the CBL_2 also reported deception on unaltered (and not deceptive) responses from the agent. In our test of 1000 cases, 20 random variable states generated false positive deception alerts at a rate of less than 1%, 42 random variable states showed false positives in between 1% and 5% of the cases, 9 random variables showed false positive results in between 5% and 10% of the cases and three showed false positive results between 10% and 20% of the cases. The K2 model only detected deceptions in states *low* and *high* successfully in less than 5% of the test cases.

Table 10. The proportion of detected deceptions.

Model	High	Normal	Low
CBL_2	0.778	0.667	0.329
$K2$	0.042	0.0	0.048
Pearson	0.286	0.0	0.0

Table 11. The rate of false activations generated.

Model	Mean	Median	Minimum	Maximum	# of states
CBL_2	0.042498296	0.02293578	0.001150748	0.821634062	74
$K2$	0.06472059	0.034682081	0.001144165	0.924050633	67
Pearson	0.053134615	0.0535	0.002	0.12	52

5 Conclusions

In this paper we have explored some techniques for implementing the first step in deception detection, namely Activation or recognition that something unexpected has been offered as an opinion. The results obtained indicate that, given agent opinions are correlated, it is indeed possible to predict expert opinions reasonably accurately for the purposes of detecting deception. Moreover, the novel technique of using agent correlation in a multi-agent environment performs well in our initial experiments. In the single agent (expert) case however, detecting deceptions is more difficult. The key to accomplishing this task is accurate modelling of the information source. Data mining techniques appear to offer the best overall solution to predicting expert opinions. However, a hybrid solution of several models constructed with different algorithms and possibly the GroupLens algorithm may offer the best performance. Furthermore, the framework presented for deception detection may be applied independent of the chosen knowledge representation. The only requirement is that agent opinions be supplied as probability distributions over a specific set of random variables. Also, these Activation methods may be applied to stand-alone knowledge bases or to multi-agent systems.

The identification of inconsistencies in expert opinions can help identify unintentional misinformation as well as deliberate disinformation. However, it is also the case that inconsistencies between experts or apparent inconsistencies in a single expert may be due to specialized knowledge. Therefore, the detection of inconsistencies must be validated as deception by further analysis such as identifying a pattern in a deceptive opinion which matches a deception tactic and identifying a goal of the deception.

6 Future Work

Further empirical study is necessary to determine the minimum number of experts necessary for accurate detection of deceptive opinions. Further empirical measurement of detection error on crafted deceptions is needed to assist in determining the accuracy of our methods for activation of deception detection tactics. To this end we will work on a deception agent capable of crafting deceptions based on the six deception tactics of Bell and Whaley [3,12,2]. We also would like to investigate the possibility of collaborative deceit by two or more agents. It appears the model may be robust to this type of attack if there are a sufficient number of agents in the system to reduce the impact of each individual agent's opinion. Further empirical analysis is necessary to determine the threshold at which experts are not correlated enough for our method of deception detection. Further work will also consider methods for differentiating false activations from true deceptions.

Acknowledgement. This work was supported by Air Force Office of Scientific Research Grant No. F49620-03-1-0014.

References

1. I. Beinlich, G. Suermondt, R. Chavez, and G. Cooper. The alarm monitoring system. In *Second European Conference on AI and Medicine*, 1989.
2. J. Bowyer Bell and Barton Whaley. *Cheating and Deception.* Transaction Publishers, 1991.
3. J. Barton Bowyer. *Cheating.* St. Martin's Press, 1982.
4. John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. Technical Report MSR-TR-98-12, Microsoft Research, Microsoft Corporation, One Microsoft Way, Redmond, WA, 98052, October 1998.

5. J. K. Burgoon and D. B. Buller. Interpersonal Deception: III. Effects of Deceit on Perceived Communication and Nonverbal Behavior Dynamics. *Journal of Nonverbal Behavior*, 18(2):155–184, 1994.
6. Daniel C. Denett. *The Intentional Stance*. The MIT Press, 1987.
7. Joey F. George and John R. Carlson. Group support systems and deceptive communication. In *Proceedings of the 32nd Hawaii International Conference on System Sciences*. IEEE, January 1999.
8. Paul Johnson and Stefano Grazioli. Fraud Detection: Intentionality and Deception in Cognition. *Accounting, Organizations and Society*, 25:355–392, 1993.
9. Eugene Santos Jr. and Gregory Johnson Jr. Toward detecting deception in intelligent systems. To appear in Proceedings of the SPIE Defense and Security Symposium 2004, April 2004.
10. Ronald Mawby and Robert W. Mitchell. Feints and Ruses: An Analysis of Deception in Sports. In Robert W. Mitchell and Nicholas S. Thompson, editors, *Deception: Perspectives on Human and Nonhuman Deceit*. State University of New York Press, 1986.
11. Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the conference on Computer supported cooperative work*, pages 175–186. IEEE, October 22-26 1994.
12. Barton Whaley. Toward a General Theory of Deception. In J. Gooch and A. Perlmutter, editors, *Military Deception and Strategic Surprise*. Frank Cass, 1982.

A Decision-Theoretic Graphical Model for Collaborative Design on Supply Chains

Y. Xiang*, J. Chen*, and A. Deshmukh†

* University of Guelph, Canada
† University of Massachusetts at Amherst, USA

Abstract. We propose a decision-theoretic graphical model for collaborative design in a supply chain. The graphical model encodes the uncertain performance of a product resultant from an integrated design distributively. It represents preference of multiple manufacturers and end-users such that a decision-theoretic design is well-defined. We show that these distributed design information can be represented in a multiply sectioned Bayesian network. This result places collaborative design in a formal framework so that it can be subject to rigorous algorithmic study.

1 Introduction

Supply chain literature has focused mostly on supply chain management [4]. This work emphasizes collaborative design in a supply chain. We consider component-centered design in which a final product is designed as a set of components supplied by manufacturers in a supply chain. Hence, the manufacturers are collaborative designers. We interpret design under broad design-for-X (DFX) concepts including design for assembly, manufacture, disassembly, environment, recyclability, etc, with the objective of producing an overall optimal performance.

From the management perspective, a supply chain consists of business entities such as customers, retailers, manufacturers, and suppliers [4]. We adopt an abstraction of the supply chain from the design perspective: The supply chain includes only entities directly involved in the collaborative design and each is referred to as a *manufacturer*. In the simplest case, a supply chain produces a single final product t and we abstract all entities not directly involved in the design (e.g., retailers, distributors and customers) of t as the *end user*. We regard the end user as outside the supply chain.

Such a supply chain can be modeled by a directed acyclic graph (DAG). Each node corresponds to a manufacturer. Each arc corresponds to a supplying relation and is directed from a *consumer* to a *supplier* (both are manufacturers). The role of consumer or supplier is relative, depending on which supplying relation is in question. A manufacturer is a consumer relative to an outgoing arc and is a supplier relative to an incoming arc. The root (node without parent nodes) R of the supply chain is the manufacturer of the final product, which supplies the product to the end user. The leaves are suppliers who are not consumers in the supply chain.

Contemporary practice of design in a supply chain is either centralized or essentially top-down [4]. In a top-down design, R designs the product by decomposing it into components. For each component C to be supplied to R, a supplier S further decomposes C into subcomponents, and the process continues. In summary, consumers play dominant roles in the design and suppliers are more passive. Such consumer-dominant designs are unlikely to be optimal because consumers are unlikely to be in the best position to judge the options available to suppliers. We propose a computational framework for collaborative design where suppliers play equally active roles in shaping the final design.

Given a design, the actual performance of the final product is uncertain due to the uncertainties in raw materials used, in the actual manufacturing process, and in the operating environment. Conventional approach of deterministic design is undesirable because it assumes representative maximum loads and minimum material property, which often leads to overdesign and inability to risk evaluation. Probabilistic approach optimizes design in the face of these uncertainties [2, 1]. We extend the probabilistic approach to a decision-theoretic approach which incorporates explicitly the preference of manufactures and end users, and to collaborative design which requires distributed reasoning.

Many of the objectives of this work have been articulated in the literature informally, e.g., [7, 4]. The key contribution of this work is the proposal of a formal decision-theoretic graphical model that explicitly encodes the information on supply chain organization, design constraints, performance measures, utilities of manufactures and end users. Such model provides a rigorous multiagent framework for computational study of collaborative design and for distributed decision aids to collaborative designers.

In Section 2, we define the problem of decision-theoretic design in a centralized context, which is expressed as a graphical model in Section 3. In Section 4, we cover the background on multiply sectioned Bayesian networks (MSBNs) as a graphical model for multiagent inference. We extend MSBNs into a knowledge representation formalism for multiagent collaborative design in Section 5. We outline the research issues and future work in Section 6.

2 Decision-Theoretic Design

A product has a *design space* described by a set D of variables. Each variable in D is called a *design parameter*. For example, the type of CPU used in a smart appliance is a design parameter and the height of the case is another. As mentioned in Section 1, we interpret design under broad concepts of DFX and hence design parameters are interpreted broadly as well. In this paper, we assume that each design parameter is associated with a discrete domain of possible values and a naturally continuous parameter is discretized. A *(complete) design* is an assignment of values to all variables in D and a *partial design* is an assignment of values to variables in a proper subset of D.

An assignment of values to variables in D must satisfy a set of constraints. Otherwise, the design cannot be realized. For instance, a computer design with a case of length L and a motherboard of length $L' > L$ cannot be realized. A constraint involves a subset $S \subset D$ of variables and specifies the allowable combinations of values for S [10]. A design (partial or complete) is *invalid* if it violates one or more constraints. Otherwise, the design is *valid*.

Different designs result in products with different performances. Given a design with a given performance, it may or may not be desirable to different people. We use the term *performance* to refer to *objective* measures of the functionalities of a product resultant from a design. That is, the value of a performance measure is independent of who is making the measurement. For example, the maximum speed of a car is a performance measure. For simplicity, we refer to the performance of a product resultant from a design as the performance of the design. The *performance space* of a product is described by a set M of variables. Each variable in M is called a *performance measure.* In this paper, we assume that each performance measure is associated with a discrete domain and a naturally continuous measure is discretized.

The performance of a product also depends on the environment in which it operates, for instance, high level of humidity may cause a digital system to fail. We describe such environmental factors by a set T of discrete variables. For each $t \in T$, a probability distribution quantifies the uncertainty over its possible values. Hence, formally, each performance measure is a function of a subset of $D \cup T$.

A principal's *subjective* preference of a design is represented by a *utility function* [6]. We assume that the utility of a design is directly dependent on the performance of the corresponding product. That is, the utility does not depend on directly on design parameters. This is mainly for conceptual and representational clarity. For any practical situation where the utility directly depends on some design parameters, it is always possible to introduce one or more performance measures to mediate the dependency. Formally, we denote the utility function as $U(M)$.

For simplicity of representation and acquisition, we assume that the utility function $U(M)$ satisfies the *additive independence* condition [6] and can be decomposed as follows: First, the performance measures are partitioned into groups $M_0, M_1, ...$ Each group M_i is associated with a utility function $U_i(M_i)$ whose value is normalized to $[0, 1]$. The overall utility function $U(M)$ satisfies $U(M) = \sum_i k_i \; U_i(M_i)$, where each weight $k_i \in (0, 1)$ and $\sum_i k_i = 1$.

Due to the uncertainty in the performance of the product given its design, we can only evaluate the expected utility of a design. Denote a given design by $D = \mathbf{d}$. Denote one possible combination of performance values of the resultant product by $M = \mathbf{m}$. Then the probability $P(\mathbf{m}|\mathbf{d})$ expresses the likelihood of performance $\mathbf{m}$ of the product resultant from design $\mathbf{d}$. The expected utility of $\mathbf{d}$ relative to utility $U_i()$ is $EU_i(\mathbf{d}) = \sum_{\mathbf{m}} U_i(\mathbf{m}) \; P(\mathbf{m}|\mathbf{d})$, where $U_i(\mathbf{m})$ is evaluated according to $U_i(M_i)$ with the values in $\mathbf{m}$ for variables outside M_i dropped. The expected utility of design $\mathbf{d}$ is then $EU(\mathbf{d}) = \sum_i k_i \; (\sum_{\mathbf{m}} U_i(\mathbf{m}) \; P(\mathbf{m}|\mathbf{d}))$. The problem of a decision-theoretic design given (D, T, M, U) can be specified as to find a valid design $\mathbf{d}^*$ that maximizes $EU(\mathbf{d})$ among all valid designs.

3 Graphical Model Representation

In order to compute $EU(\mathbf{d})$ effectively, we represent the design problem as a graphical model, which we term as a *design network.* The structure of the network is a DAG $G = (V, E)$. The set of nodes is $V = D \cup T \cup M \cup U$, where, without confusion, we overload the notation U to denote a subset of nodes each corresponding to a utility function $U_i()$.

Each arc in E is denoted (p, c) directed from the parent node p to the child node c. As V consists of four types of nodes, there are 16 potential types of arcs. We define the following 5 types as *legal* arcs:

1. Arc (d, d') between design parameters $d \in D$ and $d' \in D$.

   The arc signifies that the two parameters are involved in a constraint.

2. Arc (d, m) from a design parameter $d \in D$ to a performance measure $m \in M$.

   The arc signifies that the performance measure m depends on the design parameter d.

3. Arc (t, m) from an environment variable $t \in T$ to $m \in M$.

4. Arc (m, m') between performance measures $m \in M$ and $m' \in M$.

   The arc signifies that m' is a composite performance measure function. Because m is also a performance measure function which may or may not be composite, all root ancestors of m must be members of D or T. Fig. 1 (a) illustrates such a case, where $m' = f(m(d_1, d_2), d_3, t)$.

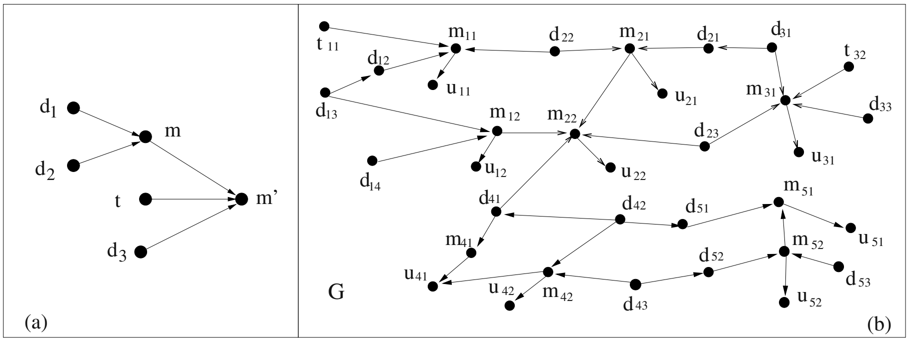


Fig. 1. (a) Performance measure m' as a composite function. (b) Graphical model of a design network.

5. Arc (m, u) from a performance measure $m \in M$ to a utility node $u \in U$.

   The arc signifies that the utility u depends on the performance measure m.

The other 11 types of arcs are (m, d), (u, d), (u, m), (d, t), (t, d), (t, t'), (m, t), (t, u), (u, t), (d, u) and (u, u'). The first 9 of them are inconsistent with the interpretation of environment factor, performance measure and utility. Thus they are illegal. Arc (d, u) is regarded illegal for reasons of clarity as explained in Section 2. Arc (u,u) allows arbitrary integration of utilities and enables specification of non-additive utilities. It is a source of potential violation of the additive independence assumption discussed in Section 2. We therefore regard such arcs as illegal at the current stage of this research. Fig. 1 (b) shows a graphical model for design.

Each node in the design network is associated with a numerical distribution of the corresponding variable conditioned on its parent variables. The distribution at a design parameter encodes a design constraint and that at an environment variable encodes uncertainty on operating condition. The distribution at a performance measure encodes a performance function and that at a utility node encodes a utility function. We propose the following representation such that each distribution is *syntactically* a conditional probability distribution. Such uniform representation allows design computation to be conducted using existing inference algorithms for probabilistic networks [9, 5, 8, 11].

Consider first a constraint over a subset $X \subset D$ of design parameters. Jensen [5] proposed a probabilistic encoding of a constraint over a set X of variables by introducing an additional binary variable $c \in \{y, n\}$ with its parent set X. For each combination $\mathbf{x}$ of values for variables in X, if it is allowable according to the constraint, then $P(c = y|\mathbf{x})$ is assigned the value 1. Otherwise, it is assigned the value 0. The distribution $P(c|X)$ has the space complexity of $O(2^{|X|+1})$.

We propose an alternative representation that is more compact. Consider a constraint over 10 binary variables $d_0, ..., d_9$. Suppose that all combinations of $d_0, ..., d_8$ are allowed. However, for some combinations of $d_0, ..., d_8$, some values of d_9 are not allowed. We represent this constraint by assigning $d_0, ..., d_8$ as the parents of d_9. For each combination of values of $d_0, ..., d_9$, if it is allowable, we assign $P(d_9|d_0, ..., d_8) = 1$. Otherwise, we assign $P(d_9|d_0, ..., d_8) = 0$. The size of $P(d_9|d_0, ..., d_8)$ is then $2^{10} = 1024$ instead of $2^{11} = 2048$ as the alternative representation.

It is possible that not all combinations of $d_0, ..., d_8$ are allowed either. In general, for any subset of two or more elements, certain combinations of values may be disallowed. The following general representation can be used: Let d_0 be the parent of d_1 and let $P(d_1|d_0)$ be assigned similarly as above. Then, Let d_0, d_1 be the parent of d_2 and let $P(d_2|d_0, d_1)$ be assigned. Repeat the process until $P(d_9|d_0, ..., d_8)$ is assigned. For each d_i above, if no value of d_i is disallowed for any allowable combination of $d_0, ..., d_{i-1}$, then the corresponding step can be skipped. This provides the best case space complexity of 1024 and the worst case complexity of $2^2 + 2^3 + ... + 2^{10} = 2044$. The alternative representation has the constant space complexity of 2048. Our method does not need to introduce additional nodes.

It follows from the types of legal arcs that all root nodes of G are elements of either D or T. Each root node is associated with a unconditional probability distribution. $P(d)$ for $d \in D$ suggests the most commonly used values of d, given no other information, and $P(t)$ for $t \in T$ reflects the uncertain environment condition.

For each node $m \in M$ whose parents $X \subset D \cup T$, $P(m|X)$ is a typical probability distribution representing the likelihood of performance values given partial designs over X. If m is a composite function, in which case, at least one of its parent m' is an element of M, $P(m|X)$ is interpreted similarly by taking into account $P(m'|X')$, where X' is the parents of m'. For the example in Fig. 1 (a), $P(m|d_1, d_2)$ and $P(m'|m, d_3, t)$ together define the likelihood of performance values of m' given partial designs over d_1, d_2, d_3 with environment condition t.

For each utility function $U_i(M_i)$, there is a utility node u_i in G with its parent set being M_i. We assign the corresponding variable u_i a binary domain $\{y, n\}$. We assign the distribution at u_i as $P(u_i = y|M_i) = U_i(M_i)$. We specify $P(u_i = n|M_i) = 1 - P(u_i = y|M_i)$. Thus, $P(u_i|M_i)$ is a syntactically valid probability distribution but semantically encodes the utility function $U_i(M_i)$.

From the above specification of the design network, it is a simple matter to verify that in its structure G, graphical separation (strictly speaking, d-separation [9]) corresponds to conditional independence. Since the design network encodes the probabilistic dependence between design and performance as well as the utility of a principal, it is semantically a *value network*. On the other hand, syntactically, the design network is a Bayesian network [9]. It is straightforward to show that $P(D)$ can be obtained by the product of distributions associated with nodes in D and it represents the likelihood of each potential design. $P(D, M)$ can be obtained by the product of distributions associated with nodes in $D \cup M$ and it represents the likelihood of each valid design and each possible performance of the design.

Suppose we perform a standard probabilistic inference in the above specified network with a valid design $\mathbf{d}$ entered into the corresponding nodes. After belief propagation, $P(M_i|\mathbf{d})$ can be obtained. At node u_i, we have

$$\begin{aligned}P(u_i = y|\mathbf{d}) &= \textstyle\sum_{\mathbf{m_i}} P(u_i = y|\mathbf{m_i}, \mathbf{d})P(\mathbf{m_i}|\mathbf{d}) = \sum_{\mathbf{m_i}} P(u_i = y|\mathbf{m_i})P(\mathbf{m_i}|\mathbf{d}) \\ &= \textstyle\sum_{\mathbf{m_i}} U_i(\mathbf{m_i})P(\mathbf{m_i}|\mathbf{d}) = EU_i(\mathbf{d}).\end{aligned}$$

The first equation above is the probabilistic inference known as *reasoning by case*. The second equation is due to the independence encoded in the network. In this case, u_i is graphically separated from design parameters by the parent nodes of u_i, namely, M_i. This equation represents the normal inference computation at node u_i during belief propagation. The third equation holds due to the assignment of the distribution to u_i. Hence, $EU_i(\mathbf{d})$ can be obtained at the node u_i after standard belief propagation. Similar idea is explored by Cooper [3] in the context of decision making in influence diagrams by using probabilistic inference.

In addition to the distribution $P(u_i|M_i)$ associated with each utility node u_i, we also associate u_i with the weight k_i (see Section 2). Hence, by integrating $EU_i(\mathbf{d})$ for all i, $EU(\mathbf{d})$ can be obtained for a given design $\mathbf{d}$ after standard probabilistic inference in the design network.

4 Overview of Multiply Sectioned Bayesian Networks

In this section, we briefly review the background on multiagent distributed probabilistic reasoning using a representation known as Multiply Sectioned Bayesian Networks (MSBNs) [11]. In the next section, we show how to represent knowledge for collaborative design in a supply chain as an MSBN. Our choice using

MSBNs is deeply rooted: From a few high level requirements, namely, (1) exact probabilistic measure of agent belief, (2) agent communication by belief over small sets of shared variables, (3) a simpler agent organization, (4) DAG domain structuring, and (5) joint belief admitting agents' belief on private variables and combining their beliefs on shared variables, it has been shown [12] that the resultant representation of a cooperative multiagent system is an MSBN or some equivalent.

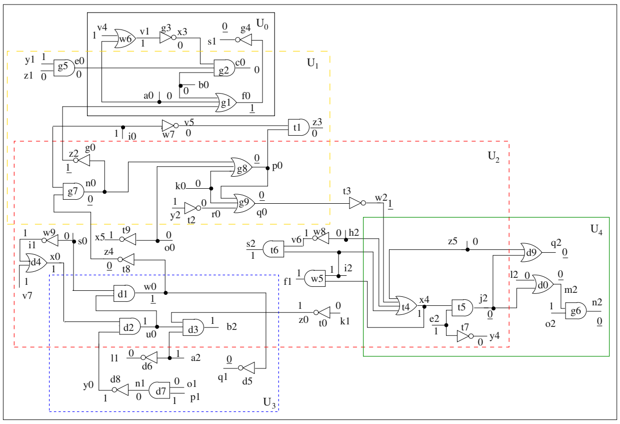


Fig. 2. A digital system consisting of 5 components.

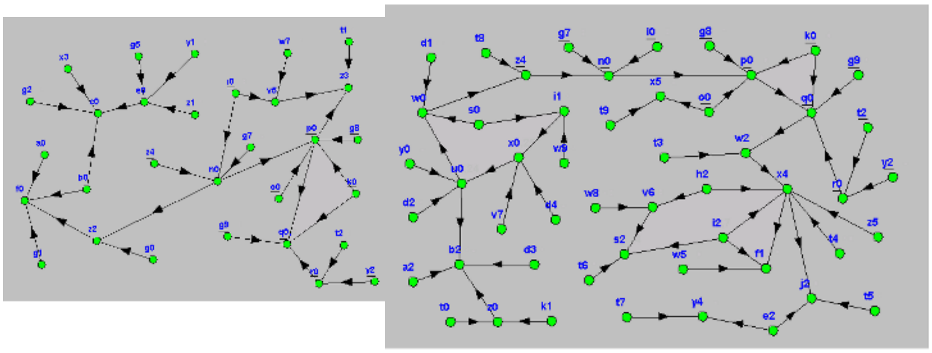


Fig. 3. Left: The subnet G_1 for U_1. Right: The subnet G_2 for U_2.

A Bayesian Network (BN) [9] S is a triplet (V, G, P) where V is a set of domain variables, G is a DAG whose nodes are labeled by elements of V, and P is a joint probability distribution (jpd) over V, specified in terms of a distribution for each variable $x \in V$ conditioned on the parents $\pi(x)$ of x in G. An MSBN M is a collection of Bayesian subnets that together define a BN. The most well-studied application of MSBNs is diagnosis and monitoring and we will use such an example to illustrate: Fig. 2 shows a piece of digital equipment made out of five components U_i $(i = 0, ..., 4)$.

Each box in the figure corresponds to a component and contains the logical gates and their connections with the input/output signals of each gate labeled.

A set of five agents, A_i ($i = 0, ..., 4$), cooperate to monitor the system and trouble-shoot it when necessary. Each agent A_i is responsible for a particular component U_i. The knowledge of an agent about its assigned component can be represented as a BN, called a *subnet*. The subnet for agent A_1 is shown in Fig. 3 (left) and that for A_2 is shown in the right. Each node is labeled with a variable name. Only the DAG of the subnet is shown in the figure with the conditional probability distribution for each variable omitted. The five subnets (one for each component) collectively define an MSBN, which form the core knowledge of the multiagent system. Based on this knowledge and limited observations, agents can cooperate to reason about whether the system is functioning normally, and if not, which devices are likely to be responsible.

To ensure correct, distributed probabilistic inference, subnets in an MSBN are required to satisfy certain conditions. To describe these conditions, we introduce the terminologies first. Let $G_i = (V_i, E_i)$ ($i = 0, 1$) be two graphs (directed or undirected). G_0 and G_1 are said to be *graph-consistent* if the subgraphs of G_0 and G_1 spanned by $V_0 \cap V_1$ are identical. Given two graph-consistent graphs $G_i = (V_i, E_i)$ ($i = 0, 1$), the graph $G = (V_0 \cup V_1, E_0 \cup E_1)$ is called the *union* of G_0 and G_1, denoted by $G = G_0 \cup G_1$. Given a graph $G = (V, E)$, a partition of V into V_0 and V_1 such that $V_0 \cup V_1 = V$ and $V_0 \cap V_1 \neq \emptyset$, and subgraphs G_i of G spanned by V_i ($i = 0, 1$), G is said to be *sectioned* into G_0 and G_1. See Fig. 4 for an example. Note that if G_0 and G_1 are sectioned from a third graph, then G_0

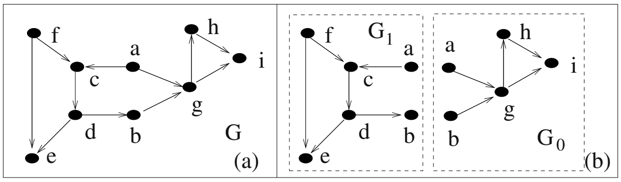


Fig. 4. G in (a) is sectioned into G_0 and G_1 in (b). G is the union of G_0 and G_1.

and G_1 are graph-consistent. The union of multiple graphs and the sectioning of a graph into multiple graphs can be similarly defined.

Graph sectioning is useful in defining the dependence relation between variables shared by agents. It is used to specify the following hypertree condition which must be satisfied by subnets in an MSBN:

Definition 1 *Let $G = (V, E)$ be a connected graph sectioned into subgraphs $\{G_i = (V_i, E_i)\}$. Let the subgraphs be organized into an undirected tree Ψ where each node is uniquely labeled by a G_i and each link between G_k and G_m is labeled by the non-empty* `interface` *$V_k \cap V_m$ such that for each i and j, $V_i \cap V_j$ is contained in each subgraph on the path between G_i and G_j in Ψ. Then Ψ is a* `hypertree` *over G. Each G_i is a* `hypernode` *and each interface is a* `hyperlink`.

Fig. 5 illustrates a hypertree for the digital system, where G_1 and G_2 are shown in Fig. 3.

The hypertree represents an organization of agent communication, where variables in each hypernode are local to an agent and variables in each hyperlink are shared by agents. Agents communicate in an MSBN by exchanging their beliefs over shared variables. We use *nodes* and variables interchangeably when

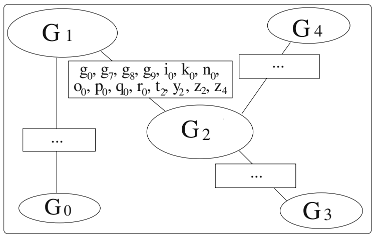

G 1
G 4
$g_0, g_7, g_8, g_9, i_0, k_0, n_0, o_0, p_0, q_0, r_0, t_2, y_2, z_2, z_4$
...
G 2
...
...
G 0
G 3

Fig. 5. The hypertree for the digital equipment monitoring system.

there is no confusion. Nodes shared by subnets in an MSBN must form a *d-sepset*, as defined below:

Definition 2 *Let G be a directed graph such that a hypertree over G exists. A node x contained in more than one subgraph with its parents $\pi(x)$ in G is a* `d-sepnode` *if there exists at least one subgraph that contains $\pi(x)$. An interface I is a* `d-sepset` *if every $x \in I$ is a d-sepnode.*

The interface between G_1 and G_2 contains 13 variables indicated in Fig. 5. The corresponding nodes in Fig. 3 are underlined. It is a d-sepset because these variables are only shared by G_1 and G_2, and each variable has all its parents contained in one of them. For instance, the parents of z_4 (t_8 and w_0) are all contained in G_2, while those of n_0 (i_0, g_7 and z_4) are contained in both G_1 and G_2 (see Fig. 3). The structure of an MSBN is a multiply sectioned DAG (MSDAG) with a hypertree organization:

Definition 3 *A* `hypertree MSDAG` *$G = \bigcup_i G_i$, where each G_i is a DAG, is a connected DAG such that (1) there exists a hypertree Ψ over G, and (2) each hyperlink in Ψ is a d-sepset.*

Note that although DAGs in a hypertree MSDAG form a tree, each DAG may be multiply connected: A *loop* in a graph is a sequence of nodes $a, b, c, ..., a$ such that the first node is identical to the last node and there is a link (not necessarily in the same direction) between each pair of nodes adjacent in the sequence. Such a loop is also referred to as an *undirected loop.* A DAG is *multiply connected* if

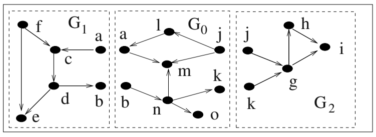


Fig. 6. A MSDAG with multiple paths across local DAGs.

it contains at least one (undirected) loop. Otherwise, it is singly connected. For example, G_1 in Fig. 3 has two loops. One of them is $(i_0, v_5, z_3, p_0, n_0, i_0)$. Hence,

G_1 is multiply connected. G_2 has several loops and is also multiply connected. Moreover, multiple paths may exist from a node in one DAG to another node in a different DAG after the DAGs are unioned. For instance, in Fig. 6, there are several (undirected) paths from node c in G_1 to node g in G_2. There is one path going through nodes a, l and j and another path goes through d, b, n and k. Each path goes across all three DAGs.

An MSBN is then defined as follows. Uniform potentials (constant distributions) are used to ensure that quantitative knowledge about the strength of dependence of a variable on its parent variables will not be doubly specified for the same variable.

Definition 4 *An MSBN M is a triplet $(V, G, \mathcal{P})$. $V = \bigcup_i V_i$ is the* `domain` *where each V_i is a set of variables. $G = \bigcup_i G_i$ (a hypertree MSDAG) is the* `structure` *where nodes of each DAG G_i are labeled by elements of V_i. Let x be a variable and $\pi(x)$ be all the parents of x in G. For each x, exactly one of its occurrences (in a G_i containing $\{x\} \cup \pi(x)$) is assigned $P(x|\pi(x))$, and each occurrence in other DAGs is assigned a uniform potential. $\mathcal{P} = \prod_i P_i$ is the* `jpd`, *where each P_i is the product of the potentials associated with nodes in G_i. A triplet $S_i = (V_i, G_i, P_i)$ is called a* `subnet` *of M. Two subnets S_i and S_j are said to be adjacent if G_i and G_j are adjacent on the hypertree MSDAG.*

MSBNs provide a framework for reasoning about an uncertain domain in cooperative multiagent systems. Each agent holds its partial perspective (a subnet) of a domain, reasons about the state of its subdomain with local observations and through limited communication with other agents. Each agent may be developed by an independent developer and the internals of an agent (agent privacy) are protected. Using the inference algorithms of MSBNs [11], agents can acquire observations in parallel and reason distributively, while their beliefs are exact relative to an equivalent centralized system.

5 Knowledge Representation for Collaborative Design

In this section, we extend the design network for decision-theoretic design presented in Section 3 to a *collaborative design network* for design in a supply chain. Recall that a design network is semantically a value network and syntactically a Bayesian network. It will be seen that a collaborative design network is semantically a distributed value network and syntactically an MSBN.

Using the MSBN representation, design knowledge is distributed among subnets. Except for public nodes (shared by two or more subnets), the knowledge about each subdomain is specified by the corresponding manufacturer and is owned privately. Therefore, a collaborative design network forms a cooperative multiagent system. Below, we consider knowledge representation issues in this social context.

5.1 How to Generate the Hypertree Structure

We assume that there are n manufacturers in the supply chain and the supply chain has a topology of a DAG where each arc is directed from a consumer to a supplier.

A directed acyclic graph may be singly connected (a unique path exists between any two nodes) or multiply connected. If it is multiply connected, there must be a undirected cycle and a manufacturer R on the cycle such that R is the supplier of two consumers C_1 and C_2 both of which are on the cycle and are adjacent to R. This dual-supplying relation has two possible cases: either R supplies two distinct components one to each consumer or R supplies identical components to both. If the components are distinct, they usually require different design and production processes and involve different departments within the organization of R. We can then treat each department as a separate supplier. That is, we replace R and represent the two departments as R_1 who supplies to C_1 and R_2 who supplies to C_2. This breaks the cycle at R.

It is also possible that R supplies the same component to each consumer. In this case, we treat the number of components as a design parameter (with a preferred value of 2) and pretend that R supplies to only one of the two consumers. The consequence is that the cycle at R is broken by deleting the arc going from one of the consumers into R.

If we apply the above technique to each cycle in the supply chain graph, it will eventually become singly connected. Fig. 7 shows an example. The singly

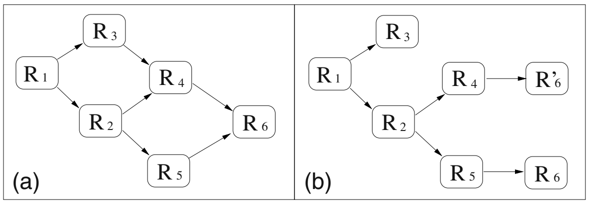


Fig. 7. (a) Supply chain graph where R_4 supplies the same type of components to R_2 and R_3, and R_6 supplies different components to R_4 and R_5. (b) After removing cycles.

connected supply chain graph becomes the basis for the hypertree of the MSBN. Each manufacturer in the supply chain is in charge of the representation of a subnet, the local computation within the subnet, and the necessary communication with the adjacent subnets. Without confusion, we will use the terms *agent* and *manufacturer* interchangeably.

In particular, each agent A_i is associated with a subnet $S_i = (G_i, P_i)$, where $G_i = (V_i, E_i)$ is the graphical subnet structure and P_i is the set of conditional probability distributions defined over V_i. Note that in general, each V_i contains a subset of D, a subset of M, and a subset of U, as defined in a design network.

We emphasize that after cycle removal the resultant supply chain graph does not have to be a directed tree such as the one in Fig. 7 (b). A directed singly connected graph can have multiple root nodes. Each root node represents a manufacturer of a final product. A supply chain graph with multiple root nodes thus represents a supply chain that produce multiple related final products. Our representation is sufficiently general to accommodate such practice.

Our key observation is that with the modification discussed above, a supply chain graph is isomorphic with a hypertree, and the dependence structure of a supply chain can be modeled as an MSDAG. Fig. 8 illustrates a collaborative design network structure which is equivalent to the centralized design network in Fig. 1 (b).

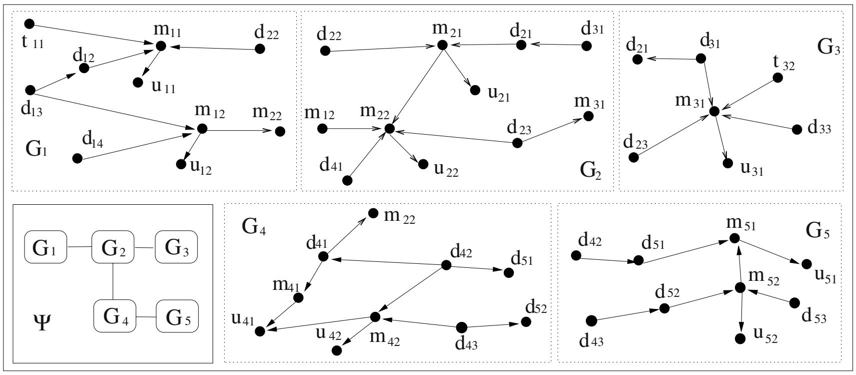


Fig. 8. Graphical model of a collaborative design network, where Ψ is the hypertree.

5.2 Partition of Design Responsibility

In a supply chain, each manufacturer is responsible to design a component to be supplied. As components must interfact with each other, certain design parameters will affect multiple components. For example, if an AC adaptor provides the power for several electronic devices, then the output voltage of the adaptor will affect the design of these devices. In general, we assume design parameters such as this are public variables and will be included in the agent interface.

When a design parameter is public, it would cause conflict if each agent that shares the parameter assigns to it a distinct value. To avoid such conflict, the value assignment authority should be partitioned among agents. That is, each agent A_i will be assigned a subset $D_i \subset D$ and A_i will be the authority to determine the partial design over D_i. The subsets satisfies $D_i \cap D_j = \emptyset$ for any distinct i and j and $\cup_i D_i = D$. In other words, $\{D_0, D_1, ...\}$ is a partition of D. We will refer to it as the *partition of design authority*. The partition can be specified permanently or can be formulated dynamically over the lifetime of the supply chain. For now, we assume a permanent partition.

In particular, we define the partition of design authority as follows:

1. For each private variable (contained in a single subnet) $d \in V_i$, agent A_i has the design authority over d, i.e., $d \in D_i$.
2. For each public variable d', one agent, say A_j, that shares d' is arbitrarily selected such that $d' \in D_j$.

Note that in practice, for each public variable d, there often exists an agent that is natural to assume the responsibility to design d. For example, the agent who designs the AC adaptor is a natural candidate to determine the value for its output voltage. In such a case, the natural candidate should be given the design authority. We stipulated above that A_j is selected arbitrarily. It simply means that the proposed framework does not dictate this choice.

5.3 How to Obtain Numerical Distributions

Numerical information encoded in a design network includes the following:

1. Unconditional probability distribution for each root design parameter node.
2. Conditional probability distribution for each non-root design parameter node.
3. Unconditional probability distribution for each root environment node.
4. Conditional probability distribution for each performance measure node.
5. Utility distribution for each utility node.
6. Weight for each utility node.

Using the MSBN representation, these distributions are distributed in subnets. Except for public nodes, these distributions are specified by the corresponding manufacturers and owned privately. Below, we consider each type of distributions in this social context.

For distributions associated with root design nodes, they provide guidance to manufacturers on frequently used values for these design parameters. If a root node is private, the distribution is assigned by the corresponding manufacturer and reflects its past design experience. When a root node is public, the distribution can be assigned by combining the past design experience of all manufacturers who share the root node. See [11] for details on how to combine multiple agents' experience in assigning a distribution. Distributions associated with root environment nodes are handled similarly.

Distributions associated with non-root design parameter nodes represent design constraints between the nodes and their parents. If the node is private within agent A_i's subdomain, then the distribution is assigned by the corresponding manufacturer. Otherwise, manufacturers who share the node must combine their opinion in deciding the constraint and assigning the distribution.

Probability distributions at performance measure nodes encode the likelihood of potential performances given particular designs. Significant expertise and past experience are needed to assign these distributions. Often, specialized CAD software and historical databases are needed in order to assess these probabilities. If the performance measure is private, a corresponding manufacturer is responsible to assign the distribution. Otherwise, expertises from all manufacturers that share the measure can be combined.

For utility distributions, the matter is different from the above. A supply chain is a cooperative multiagent system and multiple stakeholders exist. Each manufacturer has its own short term and long term interest. The short term spans, for example, the period during which the final products of the supply chain are designed and manufactured. The long term spans, for example, the period during which the equipment needed to manufacture the final products of the current supply chain must be reused for purposes beyond what are intended by the current supply chain. Even if the short term interests of all manufacturers can be assumed the same (e.g., to design and manufacture the products with the lowest cost subject to end-user's satisfaction), their long term interests may not be common. For instance, a particular design may require manufacturer A to deploy equipment e which fits A's long term interest, but requires manufacturer B to deploy equipment e' which is against B's long term interest.

We assume that manufacturers in a supply chain are cooperative and are willing to strike a compromise. Earlier, we have assumed that the overall utility of the supply chain can be expressed as a weighted sum of multiple utility functions (the additive independence assumption). This assumption also allows a simple representation of a compromise of interests of agents in a supply chain: We require that utility nodes be private. That is, each utility node encodes the utility of the corresponding manufacturer. Some nodes may correspond to its short term interest and others correspond to its long term interest. The compromise among agents is represented by the weights k_i for summing individual utilities into the overall utility. Note that association of a weight k_i to each utility node in each subnet is a minor extension to the standard MSBN subnet representation.

Early in the paper, we excluded the end-users from the supply chain graph. However, their interest must be explicitly represented as well. We assume that the interest of each end-user is delegated by the manufacturer of the corresponding final product. This is reasonable since performance measures of the final product must be either private to the corresponding agent or shared by this agent. Therefore, the end-user's utility functions can be encoded within the agent as utility nodes that depend on the relevant performance measures.

6 Conclusion

With the hypertree structure, the subnets and its associated distributions thus defined, we term the resultant representation a *collaborative design network*. Semantically, it is a distributed value network, which encodes collaborative design space, design constraints, uncertain dependency between performance and design, and manufacturer utilities of a collaborative design process for a supply chain. Syntactically, the network is an MSBN. The design space is defined by $D = \cup_i D_i$, where nodes in each D_i are contained in one agent. The likelihood and validity of each potential design is specified by $P(D)$ - the product of distributions associated with these distributed nodes. Similarly, the likelihood of each valid design, each possible environment condition, and each possible performance of the design is specified by $P(D, T, M)$ - the product of distributions associated with nodes in $D \cup T \cup M$ distributed among agents. Using distributed probabilistic reasoning [11], $P(M_i|\mathbf{d})$ for each valid design $\mathbf{d}$ can be obtained, where M_i is a set of performance measures that share a common child utility node u_i (and hence contained in a single agent's subdomain). Using a derivation similar to that in Section 3, the overall expected utility of each design $\mathbf{d}$ reflecting the collective preference of all manufacturers and end-users of the supply chain is well-defined. In summary, a collaborative decision-theoretic design on a supply chain can be adequately modeled by a collaborative design network. Due to space limit, we are unable to include a concrete case study. We intend to do so in our future work.

To solve the collaborative decision-theoretic design problem similar to what is defined in Section 2, it amounts to the following: For each valid design, compute expected utilities of each corresponding partial design by an agent, and integrate the local results by agent communication to obtain the collective expected utility of the design. Repeat the above and determine the best design. Clearly, exhaustive search is intractable. Our ongoing research is investigating more efficient distributed algorithms.

Acknowledgement. This research is supported by NSERC Research Grant OGP0155425 and NASA Grant NCC 2-1348 and NCC 2-1353. Assistance from Zoe Jingyu Zhu is acknowledged.

References

1. S.M. Batill, J.E. Renaud, and X. Gu. Modeling and simulation uncertainty in multidisciplinary design optimization. In *The 8th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, pages 5–8, 2000.
2. K.V. Bury. On probabilistic design. *J. of Engineering for Industry, Trans of the ASME*, pages 1291–1295, Nov. 1974.
3. G.F. Cooper. A method for using belief networks as influence diagrams. In R.D. Shachter, T.S. Levitt, L.N. Kanal, and J.F. Lemmer, editors, *Proc. 4th Workshop on Uncertainty in Artificial Intelligence*, pages 55–63, 1988.
4. S.H. Huang, G. Wang, and J.P. Dismukes. A manufacturing engineering perspective on supply chain integration. In *Proc. 10th Inter. Conf. on Flexible Automation and Intelligent Manufacturing*, volume 1, pages 204–214, 2000.
5. F.V. Jensen. *An Introduction To Bayesian Networks.* UCL Press, 1996.
6. R.L. Keeney and H. Raiffa. *Decisions with Multiple Objectives.* Cambridge, 1976.
7. M. Klein, H. Sayama, P. Faratin, and Y. Bar-Yam. The dynamics of collaborative design: insights from complex systems and negotiation research. *Concurrent Engineering Research and Applications J.*, 12(3), 2003.
8. R.E. Neapolitan. *Probabilistic Reasoning in Expert Systems.* John Wiley and Sons, 1990.
9. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.
10. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 2003.
11. Y. Xiang. *Probabilistic Reasoning in Multi-Agent Systems: A Graphical Models Approach.* Cambridge University Press, 2002.
12. Y. Xiang and V. Lesser. On the role of multiply sectioned Bayesian networks to cooperative multiagent systems. *IEEE Trans. Systems, Man, and Cybernetics-Part A*, 33(4):489–501, 2003.

Feature Selection by Bayesian Networks

Estevam R. Hruschka Jr.[1], Eduardo R. Hruschka[2], and Nelson F.F. Ebecken[3]

[1]COPPE / Universidade Federal do Rio de Janeiro, Bloco B, Sala 100
Caixa Postal 68506, CEP 21945-970, Rio de Janeiro, RJ, Brasil.
estevamr@terra.com.br
[2]Universidade Católica de Santos (UniSantos), Rua Carvalho de Mendonça, 144,
CEP 11.070-906, Santos, SP, Brasil.
erh@unisantos.br
[3]COPPE / Universidade Federal do Rio de Janeiro, Bloco B, Sala 100
Caixa Postal 68506, CEP 21945-970, Rio de Janeiro, RJ, Brasil.
nelson@ntt.ufrj.br

Abstract. This work both describes and evaluates a Bayesian feature selection approach for classification problems. Basically, a Bayesian network is generated from a dataset, and then the Markov Blanket of the class variable is used to the feature subset selection task. The proposed methodology is illustrated by means of simulations in three datasets that are benchmarks for data mining methods: Wisconsin Breast Cancer, Mushroom and Congressional Voting Records. Three classifiers were employed to show the efficacy of the proposed method. The average classification rates obtained in the datasets formed by all features are compared to those achieved in the datasets formed by the features that belong to the Markov Blanket. The performed simulations lead to interesting results.

1 Introduction

Feature selection has become the focus of much research in areas in which datasets with tens or hundreds of thousands of variables[1] are available [1]. While, in a theoretical sense, having more features should only give us more discriminating power, the real-world provides us with many reasons why this is not generally the case [2]. Reunanen [3] observes that there can be many reasons for selecting only a subset of the variables: (i) it is cheaper to measure only a subset of variables; (ii) prediction accuracy might be improved through exclusion of irrelevant variables; (iii) the predictor to be built is usually simpler and potentially faster when less input variables are used; (iv) knowing which variables are relevant can give insight into the nature of the prediction problem at hand. Therefore, the problem of focusing on the most relevant information has become increasingly important [4]. In this context, feature selection techniques are very important for data mining, which is a step in the Knowledge Discovery in Databases (KDD).

[1] As in the work of Guyon & Elisseeff [1], we employ the terms *variable* (raw inputs) and *feature* (constructed inputs) without distinction.

KDD can be viewed as the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data [5]. Data mining is a step in this process that centers on the automated discovery of new facts and relationships in data, and it consists of three basic steps: data preparation, information discovery and analysis of the mining algorithm output [6]. The data preparation step involves techniques such as: data cleaning, integration, transformation, reduction, discretization and generation of concepts hierarchies [7]. Feature selection is an important method in the data reduction task [8] and it can help in all phases of the KDD process, i.e. it can select relevant features, remove irrelevant ones, improve the data quality, make mining algorithms work faster and enhance the comprehensibility of the mined results.

In a machine learning approach, the quality of the feature selection task is usually measured analyzing the classification accuracy, originating what is called a wrapper model of feature selection. In other words, wrapper methods assess subsets of variables according to their usefulness to a given predictor. These methods are often criticized, because they require massive amounts of computation [1]. In data mining applications, usually one faces very large datasets, and so this problem is aggravated. In these cases, it is interesting to employ feature selection methods called filters [8], which are usually faster than wrappers [1]. Filters select subsets of variables as a pre-processing step, independently of the chosen predictor. In other words, these methods do not use the classification results to select the most suitable features. We refer the reader interested in traditional filter methods to [1,8,9] and references therein.

In this work, we propose and evaluate a Bayesian filter. Basically, we use a Bayesian network learning method to determine the causal relationship among the features. Afterward, the features having no direct causal relationship with the class feature (outside its Markov blanket) are considered less relevant to the model and they are removed from the dataset. Koller & Sahami [2] showed that the Markov Blanket criterion only removes attributes that are really unnecessary. However, the authors also observed that finding either a true or an approximate Markov Blanket might be very hard. In their work [2], they proposed a very interesting heuristic approach to dealing with this problem. However, they recognize that their algorithm performance will be significantly improved by the use of more refined techniques, like Bayesian Networks, to choose candidate Markov Blankets. Our work contributes to this goal, and it is similar to the one used in [10]. The main difference is that in [10] a Conditional Independence Learning method is applied instead of the Heuristic Search Learning Algorithm applied in our work.

To test our filter method, we used a classification tree algorithm, a rule induction method and a Naïve Bayesian classifier. Classification trees are very popular in the data mining community because they both provide high predictive accuracy and are explainable, i.e. explicit rules can be read directly from a classification tree. Thus, it is interesting to evaluate our feature selection method in the context of classification trees. To do so, we employ a version of the popular C4.5 algorithm [11], which is provided by the WEKA System [12]. Another important aspect that should be observed is that insights gained by the user are of most interest in the majority of practical data mining applications, and this is one of machine learning's major advantages over classical statistical modeling [12]. Therefore, comprehensibility is often regarded in machine learning as the most desired characteristic of inductive methods (i.e. methods that learn from examples) [13]. Although explicit rules can be easily read directly from a decision tree, they are usually far more complex than necessary [12]. In this sense, we also evaluate our proposed feature selection approach in a rule extraction task, using the J4.8 PART method [12]. This algorithm extracts rules from pruned partial decision trees (also built using the C4.5 algorithm). Finally, we also evaluate the proposed method in the context of a Bayesian classifier, using the Naïve Bayes Method, which can be considered as a benchmark classifier.

It is important to emphasize that, in filter methods, feature selection is performed as a preprocessing step to the learning algorithm. Therefore, the bias of the learning algorithm does not interact with the bias inherent in the feature selection algorithm [2]. In this sense, our simulations consider the comparison of the Average Correct Classification Rates (ACCRs) obtained in complete datasets with those achieved in the datasets formed by the selected features. The application of the proposed method to select a subset of features is illustrated by means of simulations in three datasets (Wisconsin Breast Cancer, Mushroom and Congressional Voting Records) and the experimental results show that the features selected by means of the Markov Blanket provide good results.

Our paper is organized as follows. The next section describes the employed Bayesian feature selection method, whereas section 3 describes the simulation results. Finally, Section 4 describes the conclusions and points out some future works.

2 Bayesian Feature Selection Method

A Bayesian network is a directed acyclic graph in which the nodes represent the variables and the arcs represent a causal relationship among the connected variables. A conditional probability table gives the strength of such relationship. The variables that are not connected by an arc can be considered as having no direct causal influence on them. The strength of the causal influence is given by the conditional probability $P(X_i \mid \prod X_i)$ (X_i being the i-th variable, and $\prod X_i$ the set of parents of X_i). As it is described in [14], the conditional independence assumption (Markov condition) allows one to calculate the joint distribution of all variables as:

$$P(x_1, x_2 ,..., x_n \mid BK) = \Pi P(x_i \mid \pi_1, BK) \tag{1}$$

Where BK represents Background Knowledge. Therefore, a Bayesian network can be used as a knowledge representation that allows inferences. Learning a Bayesian network from data became an effervescent research topic in the last decade [15], and there are two main classes of learning algorithms to perform this task. The first one is based on heuristic searching methods and the second one uses the conditional independence definition to generate the network. Besides, there are also algorithms that combine these two strategies.

When using algorithms based on heuristic search, one important issue is the initial order of the dataset variables [16]. These algorithms depend on the order of the variables to determine the arcs direction such that an earlier variable is a possible parent only of the later ones. The conditional independence methods try to find the arcs direction without the need of this ordering. However, when the ordering is known the algorithms can be improved [17]. In this work, we apply a method (called $K2\chi^2$) originated from an heuristic search based learning algorithm that applies the statistical

χ^2 test to determine an initial variables order that optimizes the learning process (more details are given in section 3.1.)

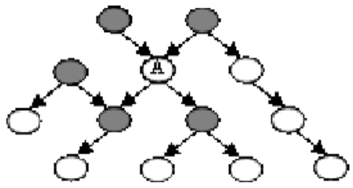


Fig. 1. The shadowed nodes represent the Markov Blanket of node *A*.

In a Bayesian network where Λ_A is the set of children of node A, and $\prod_A$ is the set of parents of node A, the subset of nodes containing $\prod_A$, Λ_A and the parents of Λ_A is called Markov Blanket of A (Figure 1). As shown in [14], the only network nodes that have influence on the conditional distribution of a given node A (given the state of all remaining nodes) are the nodes that form the Markov Blanket of A. Thus, after defining the network structure that models a dataset, the Markov Blanket of the class variable can be used as a feature subset selection criterion.

To verify the consistency of the generated network and to provide evidence that the selected features are relevant to the model, a Bayesian classification task is performed first with all the dataset features (original dataset), and later only with the selected ones (Markov Blanket features). The Bayesian feature subset selection method applied is summarized in Figure 2. In all learning tasks, a five-fold cross-validation strategy is applied. So five subsets of features (Markov Blankets) are generated and the most common one defines the selected features.

1) Learn a Bayesian network from data using K2χ^2;
2) Use a Bayesian classifier to verify the consistence of the generated network;
3) Define the Markov Blanket of the *Class* feature;
4) Remove the features outside the Markov Blanket;
5) Use a Bayesian classifier to verify the consistence of the network containing only the features inside the Markov Blanket of the class.

Fig. 2. Overview of the Bayesian feature subset selection process.

3 Simulation Results

We have employed three datasets, available at the UCI Machine Learning Repository [18], in our simulations: Wisconsin Breast Cancer, Mushroom, and Congressional Voting Records. These datasets were chosen because they are formed by ordinal, nominal and binary features respectively, showing the applicability of our method.

Section 3.1 describes K2χ^2 learning procedure as well as the simulations performed in order to verify the consistency of the obtained Bayesian networks. Section 3.2 presents the results achieved by the Bayesian networks employed for feature selection. Five-fold cross-validation is used to estimate the Average Correct Classification Rates (ACCRs) both in datasets formed by all features and in datasets formed by the selected ones. Sections 3.3, 3.4 and 3.5 describe the simulation results achieved by other classifiers. Besides, in order to compare the performance of the employed classifiers in datasets formed by all features with those formed by the selected ones, we also performed simulations considering a stratified five-fold cross-validation process, using the WEKA´s default parameters [12] for each classifier. Finally, Section 3.6 summarizes the most important simulation results.

3.1 K2χ^2 Procedure

The variable ordering is an important aspect in the bayesian network learning process, and the χ^2 statistic test can be used to assess the independence of two variables [8]. Thus, we apply the χ^2 test to optimize the variable ordering in the learning phase of the classic K2 algorithm [19]. Therefore, our method is called K2χ^2. In order to define the variables ordering, the χ^2 test is performed in each dataset variable jointly with the class feature. Thus, the strength of the dependence relationship between each variable and the class feature can be measured. Subsequently, the variables are ordered beginning with the highest χ^2 score. The first variable in the ordered list will be the one having more dependency upon the class feature.

This ordering method was applied in the three datasets to verify whether it can improve the classification rate results or not. The classification was done in a five-fold cross validation strategy, and the datasets were classified twice: i) with the original order of the variables (given in the dataset); and ii) with the order of the variables given by the χ^2 statistic. The results are given table 1.

Table 1. Correct Classification Rates applying the K2χ2 in the three datasets

Dataset	Class	K2 (original variable ordering)		K2χ2	
		Average	Std. Deviation	Average	Std. Deviation
Wisconsin Breast Cancer	0	96,84	1,66	96,61	1,58
	1	95,82	1,45	97,08	2,37
	Total	**96,48**	1,40	**96,78**	1,32
Mushroom	0	95,09	2,49	77,22	1,30
	1	5,92	3,23	87,93	1,90
	Total	**61,03**	0,72	**81,22**	0,63
Congressional Voting Records	0	63,33	28,49	96,0	5,65
	1	13,81	8,39	86,08	5,60
	Total	**40,17**	15,65	**91,40**	3,62

Table 1 shows that, in general, the K2χ^2 procedure brought better results in the three databases, so it was used in the feature selection process.

3.2 Bayesian Network Simulations

The Wisconsin Breast Cancer dataset contains examples formed by 9 features and an associated class label (benign – class 1 or malignant – class 2). The features are: clump thickness (x1), uniformity of cell size (x2), uniformity of cell shape (x3), marginal adhesion (x4), single epithelial cell size (x5), bare nuclei (x6), bland chromatin (x7), normal nucleoli (x8) and mitoses (x9). The two classes are known to be linearly inseparable. The total number of objects is 699 (458 benign, 241 malignant), of which 16 have a single missing feature. We have removed those 16 objects and used the remaining ones (444 benign and 239 malignant). The selected features were x2, x3, x4, x6, x7 and x8. The summary of the Bayesian classifier results, achieved in five-fold cross-validation, is described in Table 2 - μ and σ stand for average and standard deviation respectively. Our simulations showed that the ACCRs acquired in the complete dataset were very similar to the ones attained in the dataset formed by the selected features, leading to consistent results.

Table 2. Bayesian classifier results (%) - Wisconsin.

	All Features		Selected Features	
Class	μ	σ	μ	σ
1	96.61	1.58	96.61	1.58
2	97.08	2.37	95.40	2.70
Total	96.78	1.32	96.19	1.19

The Mushroom dataset consists of 8,124 examples, each with 22 symbolic features with up to 12 different values. 51.8% of the cases represent edible (class 1), and the rest non-edible, mostly poisonous, mushrooms (class 2). The features will be here called xi (i=1,2,…,22) according to the order that they appear in the dataset. There are 2,480 missing values (all for feature #11). These examples were removed and we applied the proposed method in the remaining ones. The selected features were x3(cap-color), x5(odor) and x9(gill-color). Table 3 provides the ACCRs, achieved in a five-fold cross-validation, considering datasets formed by all and selected features. In these simulations, the selected features even improve the Bayesian classification.

Table 3. Bayesian classifier results (%) - Mushroom.

	All Features		Selected Features	
Class	μ	σ	μ	σ
1	77.22	0.70	94.61	1.03
2	87.93	0.59	96.25	1.90
Total	81.22	0.41	95.11	0.63

The Congressional Voting Records dataset includes votes for each one of the U.S. House of Representatives Congressmen on 16 key votes. There are 435 instances (267 democrats – class 1, 168 republicans – class 2) and each one of the 16 features is Boolean valued. Besides, there are 203 instances with missing values. These instances were removed and the proposed method was employed in the 232 remaining ones (124 democrats, 108 republicans). The features will be here called xi (i=1,2…,16) according to the order that they appear in the dataset. The selected features were x3(adoption-of-the-budget-resolution), x4(physician-fee-freeze), x5(El-Salvador-aid) and x12(education-spending). The Bayesian classification results are described in Table 4, where one observes that the selected features provide very consistent results.

Table 4. Bayesian classifier results (%) - Congressional.

	All Features		Selected Features	
Class	μ	σ	μ	σ
1	96.00	5.65	96.00	5.65
2	86.08	5.60	85.08	6.25
Total	91.40	3.62	90.95	3.48

3.3 Naïve Bayes Method

The Naïve Bayes Method uses all features and allows them to make contributions to the decision that are equally important and independent of one another, given the class. This leads to a simple scheme that works well in practice [12]. Table 5 shows the results obtained in complete datasets and in datasets formed by the selected features. For the sake of brevity, in our tables *C* and *S* stand for *Complete* (formed by all features) and *Selected* (formed by selected features) datasets respectively. One can observe that the proposed method selects very predictive features, which in Mushroom and Congress provide even better ACCRs than those achieved in the dataset formed by all features.

Table 5. Average Classification Rates (%) – Naïve Bayes.

Dataset	Total	Class 1	Class 2
Wisconsin C	96.49	95.70	97.90
Wisconsin S	95.90	95.70	96.20
Mushroom C	97.36	99.60	93.80
Mushroom S	99.22	100.00	98.00
Congressional C	91.40	88.70	94.40
Congressional S	93.10	91.10	95.40

3.4 J4.8 Method

The J4.8 algorithm is WEKA´s implementation of the popular C4.5 decision tree learner [11]. In fact, J4.8 is a C4.5 improved version, called revision 8 [12]. Table 6 shows the simulation results obtained by this classifier. In this table, one can also find the features employed by the decision tree algorithm (in the first column between brackets). The features listed in this table are those used in the best classifier model, because we have employed a five-fold cross-validation process. In this sense, all the features selected by the Bayesian approach were employed both in Mushroom and in Wisconsin Breast Cancer, whereas in Congress Voting Records the feature selection process was not so important, because only one feature is necessary to classify all examples. Besides, in our simulations the selection method was consistent in the context of J4.8, and consequently with the information gain criterion.

Table 6. Average Classification Rates (%) – J4.8.

Dataset	Total	Class 1	Class 2
Wisconsin C (1,2,3,4,5,6,7)	95.17	96.40	92.90
Wisconsin S (2,3,4,6,7,8)	95.61	95.00	96.70
Mushroom C (5,18,17,8,4,20)	100.00	100.00	100.00
Mushroom S (3,5,9)	99.86	99.80	100.00
Congressional C (4)	95.26	95.20	95.40
Congressional S (4)	96.98	95.20	99.10

3.5 J4.8 PART Method

The J4.8 PART extracts rules from pruned partial decision trees built using C4.5. In other words, it combines the divide-and-conquer strategy for decision trees learning with the separate-and-conquer one for rule learning [12]. In essence, to make a single rule, a pruned decision tree is built for the current set of instances. Then, the leaf with the largest coverage is made into a rule and the tree is discarded. Table 7 shows the obtained results in the employed datasets. Again, all the selected features were employed both in Mushroom and in Wisconsin Breast Cancer datasets, whereas in Congress Voting Records only one of the selected features was enough to classify all the examples. These results are consistent with those described in the previous section and indicate that the proposed method is promising, allowing the extraction of simpler rules with very good ACCRs.

Table 7. Average Classification Rates (%) – J4.8 PART.

Dataset	Total	Class 1	Class 2
Wisconsin C (1,2,3,4,5,6,7,8)	96.05	97.50	93.30
Wisconsin S (2,3,4,6,7,8)	95.31	95.90	94.10
Mushroom C (5,8,11,17,18,20,21)	100.00	100.00	100.00
Mushroom S (3,5,9)	99.86	99.80	100.00
Congressional C (2,3,4,9,11)	94.40	95.20	93.50
Congressional S (4)	96.98	95.20	99.10

3.6 Summary of Simulation Results

In this section, we present and discuss our main simulation results. The total ACCRs are described in Table 8, where the number of features is represented between brackets. In general, the ACCRs obtained in the datasets formed by all features were very similar to the ones achieved in the datasets formed by the features selected by means of the Markov Blanket. Besides, we noticed significant improvements in relation to the number of employed features. In the Wisconsin Breast Cancer dataset, 66.67% of the original features were enough to get high classification rates. A similar effect was observed in the Mushroom dataset, where just 13.64% of the original features were selected. On top of that, the Bayesian network accuracy significantly improved. Finally, in the Congressional Voting Records dataset, with 25% of the original features one gets high classification rates. Another important aspect is that the

ACCRs provided by the employed classifiers both in the complete datasets and in the datasets formed by the selected features are comparable to the best ones found in the literature. For example, Duch et al. [13] describe classification results obtained by 15 methods. In the Mushroom dataset, one observes that these results vary from 91% to 100%, whereas in the Wisconsin they vary from 92.7% to 99%. The Congressional dataset documentation reports accuracies that vary from 90% to 95% [20].

Table 8. Main simulation results: Average Classification Rates (%) and number of features.

	Wisconsin		Mushroom		Congressional	
Classifier	C (9)	S (6)	C (22)	S (3)	C (16)	S (4)
Bayesian Network	96.78	96.19	81.22	95.11	91.40	90.95
Naive Bayes	96.49	95.90	97.36	99.22	91.40	93.10
J4.8	95.17	95.61	100.00	99.86	95.26	96.98
PART	96.05	95.31	100.00	99.86	94.40	96.98

4 Conclusions and Future Work

This work described and evaluated a feature selection approach for classification problems. We employed a Bayesian filter to define the most relevant features of a dataset. More specifically, a Bayesian network is generated from a dataset, and then the Markov Blanket of the class variable is used to the feature selection task. We illustrated the efficacy of the feature selection method by means of simulations in three datasets that are benchmarks for data mining methods: Wisconsin Breast Cancer, Mushroom and Congressional Voting Records. They are formed by ordinal, nominal and binary features respectively. Besides, the proposed method can also be employed in continuous datasets. To do so, one has to discretize the original values.

In our simulations, we compared the Average Correct Classification Rates (ACCRs) obtained in the space formed by all features with those obtained in the space formed by selected features. Our results showed that the features selected by means of the Markov Blanket are very predictive, i.e. the ACCRs provided by the employed classifiers in the datasets formed by the selected features are comparable to the best ones found in the literature. On top of that, in general the features selected by the Bayesian method were also employed by the classifiers based in the information gain criterion, what indicates the soundness of the proposed method in the context of classification trees. Since algorithms based on the information gain are very popular in the data mining community, this result is very interesting.

Our future work will focus on three main aspects. First, we are going to employ a wrapper model within the Markov Blanket. In this sense, we are going to investigate if it is possible to find a subset of the selected features, improving the knowledge structure without significantly decreasing the ACCRs. Besides, it is possible that another learning algorithm (based on conditional independence for instance) can provide a different Markov Blanket, and this verification will also be performed. Third, we are going to evaluate our methodology in other datasets.

References

[1] Guyon, I., Elisseeff, A., *An Introduction to Variable and Feature Selection*, Journal of Machine Learning Research 3, pp. 1157-1182, 2003.

[2] Koller, D., Sahami, M., *Toward optimal feature selection,* Proceedings of the 13th International Conference on Machine Learning, pp. 284-292, July, 1996.

[3] Reunanen, J., *Overfitting in Making Comparissons Between Variable Selection Methods,* Journal of Machine Learning Research 3, pp. 1371-1382, 2003.

[4] Blum, A.L., Langley, P., *Selection of Relevant Features and Examples in Machine Learning*, Artificial Intelligence, pp. 245-271, 1997.

[5] Fayyad, U. M., Shapiro, G. P., Smyth, P. From Data Mining to Knowledge Discovery : An Overview, *Advances in Knowledge Discovery and Data Mining*, Fayyad et al. Eds, pp.1-37, MIT Press, 1996.

[6] Bigus, J. P., *Data Mining with Neural Networks* , First edition, USA, McGraw-Hill, 1996.

[7] Han, J. and Kamber, M., *Data Mining, Concepts and Techniques*. Morgan Kaufmann, 2001.

[8] Liu, H. and Motoda, H., *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic, 1998.

[9] Yang, Y. and Pederson, J., *A comparative study on feature selection in text categorization*, Proc. of the Fourteenth International Conference on Machine Learning, 1997.

[10] Cheng, J., Bell, D.A., and Liu, W., *Learning belief networks from data: An information theory based approach*, Proceedings of the sixth ACM International Conference on Information and Knowledge Management, 1997.

[11] Quinlan, J. R., C4.5 Program for Machine Learning. Morgan Kaufmann, 1993.

[12] Witten, I.H., Frank, E., *Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann Publishers, USA, 2000.

[13] Duch, W., Adamczak, R., Grabczewski, K. *A New Methodology of Extraction, Optimization and Application of Crisp and Fuzzy Logical Rules*, IEEE Transactions on Neural Networks, vol.11, n.2, pp.1–31, 2000.

[14] Pearl, J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, CA., 1988.

[15] Chickering, D. M., *Optimal Structure Identification with Greedy Search,* Journal of Machine Learning Research, (3) pp. 507-554, 2002.

[16] Hruschka Jr, E. R., Ebecken, N. F. F., Ordering attributes for missing values prediction and data classification. In: *Data Mining III - Management Information Systems Series* - vol. 6, ed.Southampton - UK : WIT Press, 2002.

[17] Spirtes P., Glymour C., and Scheines R. *Causation, Predication, and Search*. Springer-Verlag, New York, 1993.

[18] Merz, C.J., Murphy, P.M., UCI Repository of Machine Learning Databases, http://www.ics.uci.edu, Irvine, CA, University of California, Department of Information and Computer Science.

[19] Cooper G. & Herskovitz, E.. A Bayesian Method for the Induction of Probabilistic Networks from Data. Machine Learning, 9, 309-347, 1992.

[20] Schllimmer, J.C., *Concept acquisition through representational adjustment,* Doctoral Dissertation, Department of Information and Computer Science, University of California, Irvine, 1987.

Radial Basis Function Network Pruning by Sensitivity Analysis

Daming Shi[1], Junbin Gao[2], Daniel So Yeung[3], Fei Chen[1],

[1] School of Computer Engineering, Nanyang Technological University, Singapore 639798
asdmshi@ntu.edu.sg

[2] School of Mathematics, Statistics and Computer Science, University of New England, Armidale, NSW 2351, Australia
jbgao@mcs.une.edu.au

[3] Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong
csdaniel@comp.polyu.edu.hk

Abstract. Radial basis function (RBF) neural networks have been extensively used for classification and regression due to the fact that they can provide fast linear algorithms to approximate any regular function. The most critical issue in the construction of an RBF network for a given task is to determine the total number of radial basis functions, their centers and widths. Conventional methods of training an RBF network are to specify the radial basis function centers by searching for the optimal cluster centers of the training examples. This paper proposes a novel learning algorithm for construction of radial basis function by sensitive vectors (SenV), to which the output is the most sensitive. Our experiments are conducted on four benchmark datasets, and the results show that our proposed SenV-RBF classifier outperforms conventional RBFs and achieves the same level of accuracy as support vector machine.

1 Introduction

A radial basis function (RBF) network provides a fast, linear algorithm capable of representing complex non-linear mappings [2], and can approximate any regular function [16]. An RBF classifier is a three-layer neural network model, in which an N-dimensional input vector $\mathbf{x}=(x_1\ x_2\ \dots\ x_N)$ is broadcast to each of K neurons in the hidden layer. Each hidden neuron produces an activation function, typically a Gaussian kernel:

$$h_i = \exp\left(-\frac{\|\mathbf{x}-\mathbf{c}_i\|^2}{2\sigma_i^2}\right), \qquad i=1,2,\dots,K, \tag{1}$$

where $\mathbf{c}_i$ and σ_i^2 are the center and width of the Gaussian basis function of the *i*th hidden unit, respectively. The units in the output layer have interconnections with all the hidden units. The *j*th output neuron has the form:

$$f_j(\mathbf{x}) = \mathbf{w}_j\mathbf{h} = \sum_{i=1}^{K} w_{ij} \exp\left(-\frac{\left\|\mathbf{x}-\mathbf{c}_i\right\|^2}{2\sigma_i^2} \right), \tag{2}$$

where $\mathbf{h}=(h_1\ h_2\ \ldots\ h_K)$ is the input vector from the hidden layer, and the w_{ij} is the interconnection weight between the *j*th output neuron and the *i*th hidden neuron.

The most critical issue in the construction of an RBF network for a given task is to determine the total number of radial basis functions along with their centers and widths. There are two different ways to specify these hidden units. One way, explicated by Bishop [2], is to cluster the training examples and then assign a neuron to each cluster. The other way, after Poggio [19], is to learn mapping from a set of examples, that is, to create a neuron for each training example and then to prune the hidden neurons by example selection.

Moody and Darken [14] located optimal set of centers using both the *k-means* clustering algorithm and learning vector quantization. This method is based on locating the dense regions of the training inputs. The centers are the means of the vectors in such regions. The drawback of this method is that it takes into consideration only the distribution of the training inputs, yet the output values influence the positioning of the centers, especially when the variation of the output in a cluster is high.

Bishop [2] introduced the Expectation-Maximization (EM) algorithm to optimize the cluster centers. Initial centers are found by clustering before applying the EM algorithm. The EM approach includes an expectation part which calculates the likelihood of the model, and a maximization part which maximizes the likelihood of the data with respect to the centers and radii.

Chen *et al.* [5] proposed orthogonal least square (OLS) learning to determine the optimal centers. The OLS combines the orthogonal transform with the forward regression procedure to select model terms from a large candidate term set. The advantage of employing orthogonal transform is that the responses of the hidden layer neurons are decorrelated so that the contribution of individual candidate neurons to the approximation error reduction can be evaluated independently. To improve the generalization and global optimization abilities of the OLS, advanced versions can be seen in [6] and [7] respectively.

Orr [15] examined regularized forward selection of centers of RBF networks. He considered the variation in the output in an optimal sense with a set of training samples selected by forward subset selection, regularization, and cross validation. Zero-order regularization along with either delete-1 or generalized cross-validation was incorporated to perform the regularized forward selection. Starting with an empty subset, one gradually selects those centers, whose contribution to reducing the error is appreciably large. This method can produce the same result as OLS.

Hwang [10] clustered training example class by class to construct an RBF network. A statistical approach called linear discrimination function is used to determine the weights. Two specific methods can be applied to obtain the linear discrimination function. One uses least maximum square error minimization procedure and the other derives the linear discriminant function under the assumption that the training patterns form a normal distribution and all the covariance matrices are the same.

Panchapakesan *et al.* [17] investigated ways to train an RBF network with a set of centers selected by unsupervised methods and then to fine tune the locations of centers. This can be done by first evaluating whether moving the centers would decrease the error and then, depending on the required level of accuracy, changing the center locations. They gave out bounds on the gradient and Hessian of the error function. If a change in centers is desired, bounds on the Hessian may be used to fix a step size, which depends on the current centers to get a guaranteed amount of decrease in the error.

Mao [12] selects RBF centers based on Fisher ratio class separability measure with the objective of achieving maximum discriminative power. The OLS is employed to decouple the correlations among the responses of the hidden units so that the class separability provided by individual RBF neurons can be evaluated independently. This method can select a parsimonious network architecture as well as centers providing large class separation.

The common feature of all the above methods is that the radial basis function centers are a set of the optimal cluster centers of the training examples. Schokopf *et al.* [21] calculated *support vectors* using a support vector machine (SVM), and then used these support vectors as radial basis function centers. Their experimental results showed that the support-vector-based RBF outperforms conventional RBFs. Although the motivation of these researchers was to demonstrate the superior performance of a full support vector machine over either conventional or support-vector-based RBFs, their idea that the *critical* vectors (training examples), rather than cluster centers, should construct the RBF for a given classification task, is worth borrowing.

This paper proposes a novel approach to determining the centers of RBF networks based on sensitivity analysis. The remainder of this paper is organized as follows: In section 2, we describe the concepts of sensitivity analysis. In section 3, the most critical vectors are obtained by OLS in terms of sensitivity analysis. Section 4 contains our experiments and Section 5 offers our conclusions.

2 Sensitivity Analysis on Neural Networks

Sensitivity is initially investigated for the construction of a network prior to its design, since problems (such as weight perturbation, which is caused by machine imprecision and noisy input) significantly affect network training and generalization. Sensitivity analysis applied to network pruning seems particularly useful and valuable when network training involves a large amount of redundant data [28].

In 1990, Stevenson established the sensitivity of Madaline to weight error and derived an analytical expression for the probability of error in Madaline [22]. However, his results are only fit for neurons with threshold activation functions and binary inputs, and they are not applicable to other continuous activation functions. In 1992, Choi and Choi presented a thorough study on sensitivity analysis of the multilayer perceptron (MLP) with differentiable activation functions [8]. They defined the sensitivity as the variant of error over the variance of the weight perturbation. Because the weights are unknown prior to network training, this method cannot be used for network design and construction purpose.

In 1995, Piche systematically discussed the selection of weight accuracies for Madaline using a statistical approach to sensitivity analysis [18]. According to his stochastic model, all neurons have the same activation function. All network inputs, weights, input perturbations, and weight perturbations are random variables. In this model, the sensitivity of Madaline is defined as the noise-to-signal (NSR) of the output layer, the output NSR of a layer of threshold Adaline is:

$$NSR = \frac{\sigma_{\Delta y}^2}{\sigma_y^2} = \frac{4}{\pi}\sqrt{\frac{\sigma_{\Delta x}^2}{\sigma_x^2} + \frac{\sigma_{\Delta w}^2}{\sigma_w^2}} \tag{3}$$

where σ_y^2, σ_x^2, σ_w^2, $\sigma_{\Delta y}^2$, $\sigma_{\Delta x}^2$ and $\sigma_{\Delta w}^2$ refer to the variances of output *y*, inputs *x*, weights *w*, output error Δy, input perturbation Δx and weight perturbation Δw, respectively. Piche's stochastic model is not generally valid because: (1) All neurons in the same layer are assumed to have the same activation function, but this is not the case in some network models. (2) To satisfy the central limit theorem, the number of neurons in hidden layers is assumed to be large. (3) Weight perturbations are assumed to be very small, but this would be too restrictive for network training.

Yeung and Sun [25] generalized Piche (1995)'s work in two significant ways: (1) No restriction on input and output perturbation, which widens the application areas of sensitivity analysis; (2) The commonly--used activation functions are approximated by a general function expression whose coefficient will be involved in the sensitivity analysis. This treatment provides a way to sensitivity analysis on activation functions. In contrast to equation (3), the sensitivity of a threshold neuron in [25] forms:

$$S = \frac{D(\hat{y} - y)}{D(y)} = \frac{4}{\pi}\sqrt{\frac{\sigma_{\Delta x}^2}{\sigma_x^2} + \frac{\sigma_{\Delta w}^2}{\sigma_w^2} + \frac{\sigma_{\Delta x}^2}{\sigma_x^2}\cdot\frac{\sigma_{\Delta w}^2}{\sigma_w^2}} \tag{4}$$

where σ_x^2, σ_w^2, $\sigma_{\Delta x}^2$ and $\sigma_{\Delta w}^2$ follow the definitions of equation (3). $D(\bullet)$ denotes variance, $\hat{y}$ and *y* denote output with and without perturbations respectively. It can be seen that, when $\sigma_{\Delta x}^2 / \sigma_x^2$ and $\sigma_{\Delta w}^2 / \sigma_w^2$ are very small, equation (3) and (4) are nearly the same. But when $\sigma_{\Delta x}^2 / \sigma_x^2$ and $\sigma_{\Delta w}^2 / \sigma_w^2$ are large, the results are different. For the threshold activation function, the output error variance cannot exceed 2. Equation (4) satisfies this, but equation (3) fails. So Yeung and Sun (2002)'s model is valid when the perturbation is either small or large.

Zeng and Yeung [26, 27] proposed a quantified measure and its computation for the sensitivity of the MLP to its input perturbation. A bottom-up approach was adopted. A single neuron is considered first, and algorithms with approximately derived analytical expressions that are functions of expected input deviation are given for the computation of its sensitivity. Then another algorithm is given to compute the sensitivity of the entire MLP network. Some applications of the MLP, such as improving error tolerance, measuring generalization ability, and pruning the network architecture, would benefit from their theoretical study. However, this method would be even more useful if it took into considerations correlation among training examples.

Some researchers studied how the real world data to produce uncertainty to neural networks. Although a different terminology is used, such an uncertainty analysis is actually the sensitivity analysis described above. Error bar estimates are usually used to allow confidence intervals to be placed around a model prediction. Wright [24] has discussed most of approaches used in dealing with the input uncertainty and proposed a Bayesian approach for this problem using the Markov Chain Monte Carlo method.

3 Sensitive Vector Learning to Construct RBF Networks

In this section, RBF classifier's sensitivity is defined as the mathematical expectation of the square of output deviations caused by the perturbation of RBF centers. An algorithm will be given that can be used to select critical vectors.

3.1 RBF Classifiers' Sensitivity to the Kernel Function Centers

We use symbols $\hat{\mathbf{c}}_i$ and $\hat{\sigma}_i$ to denote the values of center and width of the *i*th hidden neuron under a perturbation. Then the deviation resulted from this perturbation is:

$$\Delta y_j = \hat{\mathbf{w}}_j \hat{\mathbf{h}} - \mathbf{w}_j \mathbf{h} = \sum_{i=1}^{K} \hat{w}_{ij} \exp\left(-\frac{\left\| \mathbf{x} - \hat{\mathbf{c}}_i \right\|^2}{2\hat{\sigma}_i^2} \right) - \sum_{i=1}^{K} w_{ij} \exp\left(-\frac{\left\| \mathbf{x} - \mathbf{c}_i \right\|^2}{2\sigma_i^2} \right) \quad (5)$$

Here, $\hat{\mathbf{c}}_i = \mathbf{c}_i + \Delta \mathbf{c}_i$ are the centers deviated from the centers under the perturbations, and the interconnection weights under the perturbations are $\hat{\mathbf{w}}_j = \mathbf{w} + \Delta \mathbf{w}$, where $\mathbf{w}$ can be calculated using a pseudo matrix inversion, or data training. The most common method for selecting RBF widths is to make all of them equal, i.e., $\forall i$, $\sigma_i = \sigma$, and then set σ to a constant value depending on the prior knowledge of the given application. Since both the RBF widths affect classification errors but not the sensitivity of the hidden neurons, here we focus on the perturbations on the centers and their interconnection weights. The perturbation on the *i*th RBF center and the weights connected to the *j*th output, $\Delta\mathbf{c}_i$ and $\Delta\mathbf{w}_j$, can be generated following a Gaussian distribution with 0 means and variances $\sigma_{\mathbf{c}_i}$, $\sigma_{\mathbf{w}_j}$, respectively.

The RBF centers will be selected recursively in the next subsection. To make the sensitivity analysis cater for the construction of RBF networks, a recursive definition of sensitivity is given below. At the *K*th time, suppose there are a number $(K-1)$ of RBF centers fixed already, the newcomer $\mathbf{c}_i$ is observed. Hence, the *j*th output neuron's sensitivity to the current number *K* of RBF centers is defined as the mathematical expectation of $(\Delta y_j)^2$ (square of output deviations caused by the perturbations of RBF centers) with respect to all $\Delta\mathbf{c}_i$ and the training example set $D = \{\mathbf{x}_l\}_{l=1}^{L}$, which is expressed as

$$S_j^{(K)} = E[(\Delta y_j)^2] \tag{6}$$

The difference between sensitivity-based and conventional RBF networks can be illustrated in Fig. 1.

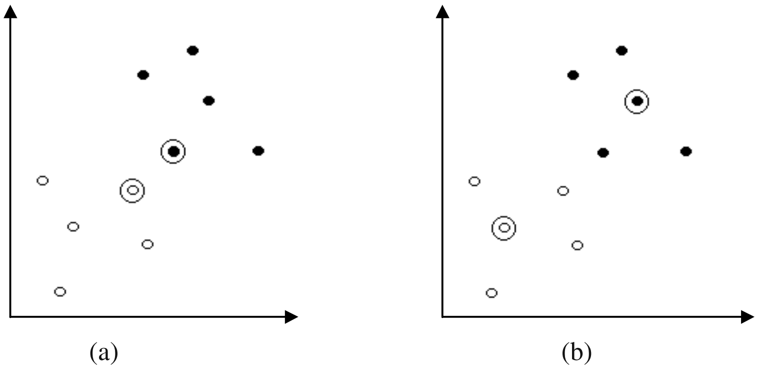


Fig. 1. Illustration of the difference between sensitivity-based and conventional RBF classifiers. The circles and balls represent data points of two classes respectively, and RBF centers are indicated by extra circles. (a) Sensitivity-based RBF, centers being the vectors sensitive to the classification. (b) Conventional RBF, centers being the cluster centroids.

3.2 Orthogonal Least Square Transform

The most critical vectors can be found by equation (7). However, the RBF centers cannot be determined by sensitivity measure only, because some of critical vectors may be correlated. The OLS [5] can alleviate this problem of redundant or correlated centers.

Let $\mathbf{Y}=(\boldsymbol{y}_1, \boldsymbol{y}_2 \dots \boldsymbol{y}_L)^T$ be the output matrix corresponding to all the number L of training examples, $\boldsymbol{y}_i$ (i=1,2,…,L) an M-dimensional vector, denoting number M of output units. We have

$$\mathbf{Y=HW=(QA)W} \tag{7}$$

Where **Y**, **H**, **W** are $L\times M, L\times L, L\times M$ matrices, respectively. The selection of RBF centers is equivalent to the selection of the most critical columns of **H**. The matrix **H** can be decomposed into **QA**, where **Q** is an $L\times L$ matrix with orthogonal columns $[\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_L]$, and **A** is an $L\times L$ upper triangular matrix as follows:

$$\mathbf{H}=\begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1L} \\ h_{12} & h_{12} & & \vdots \\ \vdots & & & \vdots \\ \vdots & & & \\ h_{1L} & \cdots & \cdots & h_{LL} \end{bmatrix}, \quad \mathbf{A}=\begin{bmatrix} 1 & a_{12} & \cdots & \cdots & a_{1L} \\ 0 & 1 & a_{23} & & \vdots \\ \vdots & 0 & \ddots & & \vdots \\ \vdots & & 0 & 1 & a_{(L-1)L} \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}. \tag{8}$$

Only one column of **H** is orthogonalized. At the Kth time, the Kth column is made orthogonal to each of the K-1 previously orthogonalized columns and the operation for K=2, … , L is repeated. The computational procedure can be represented as follows [5]:

$$\begin{cases} \mathbf{q}_1 = \mathbf{h}_1, \\ a_{ik} = \dfrac{\mathbf{q}_i^T \mathbf{h}_K}{\|\mathbf{q}_i\|}, \qquad 1 \le i < K, \\ \mathbf{q}_K = \mathbf{h}_K - \sum_{i=1}^{K-1} a_{iK}\mathbf{h}_i \end{cases} \tag{9}$$

then the RBF centers are selected by sorting these columns.

3.3 Sensitive Vector Selection

Let $\mathbf{S}^{(K)}(\mathbf{c}_i)$ denote the sensitivity of the previous (K-1) RBF centers and a candidate RBF center $\mathbf{c}_i$ which is corresponding to $\mathbf{q}_i$ at the Kth time, where $1 \le i \le L$.

Substitute any interconnection weight in equations (5) and (7) by:

$$w_{ij}^{(K)} = \sum_{l=1}^{L} a_{li} \cdot w_{ij}, \tag{10}$$

and calculate the sensitivity for all the possible K-center RBF networks. Let $\mathbf{Q}^{(K)}$ denote the orthogonal matrix at the Kth time, then the columns in $\mathbf{Q}^{(K)}$ are sorted in the order:

$$\left\|\mathbf{S}^{(K)}(\mathbf{c}_1)\right\| \geq \left\|\mathbf{S}^{(K)}(\mathbf{c}_2)\right\| \geq \cdots \geq \left\|\mathbf{S}^{(K)}(\mathbf{c}_L)\right\|. \tag{11}$$

A formal statement of the algorithm for the selection of critical vectors is given as follows:

STEP 1. Initialization. Form the matrix **H** in equation (8) by the RBF function responses of all the training examples.

STEP 2. First critical vector neuron selection. Calculate sensitivity of each column of **H** with equation (7). The column that provides the maximum sensitivity is selected as the first column of matrix $\mathbf{Q}^{(1)}$. Calculate the classification error $\mathbf{Err}^{(1)}$ with the selected RBF center.

STEP 3. Orthogonalization and critical vector selection. Let K=2.

STEP 4. Orthogonalize all remaining columns of **H** with all the columns of $\mathbf{Q}^{(K-1)}$ using equation (10).

STEP 5. Each training example, $\mathbf{c}_i$, is a candidate of the Kth RBF center, which is corresponding to the orthogonalized column $\mathbf{q}_i$, ($K \leq i \leq L$). Calculate interconnection weights using a pseudo matrix inversion and compute the sensitivity of the previous (K-1) RBF centers with each candidate center $\mathbf{S}^{(K)}(\mathbf{c}_i)$ with the weights updated by equation (11). Sort the columns in $\mathbf{Q}^{(K)}$ with equation (12), and the one yielding the maximum sensitivity is selected as the Kth column of $\mathbf{Q}^{(K)}$. Calculate the classification error $\mathbf{Err}^{(K)}$ with the selected RBF centers.

STEP 6. If ($\mathbf{Err}^{(K)}$- $\mathbf{Err}^{(K-1)}$) is smaller than a predefined threshold, go to STEP 8.

STEP 7. K++, go to STEP 4.

STEP 8. End.

The critical vectors corresponding to he first K columns in $\mathbf{Q}^{(K)}$ will be selected as hidden neurons.

4 Experiments and Results

Our experiments were conducted on 2 datasets from UCI Repository [4], and 2 datasets from the Statlog collection [13]. The experiments are described as follows:

UCI Iris Plants (*iris*). This is the best--known database in the pattern recognition literature. The dataset contains 3 classes referring to 3 types of iris plant: Setosa, Versicolour and Virginica, respectively. One class is linearly separable from the others, but the other two classes are not linearly separable from each other. There are 150 instances (50 in each class), which are described by 4 attributes, sepal length, sepal width, petal length and petal width. 5-fold cross validation is conducted on the entire data set.

UCI Glass Identification (*glass*). This study is motivated by criminological investigation using the glass as evidence at the scene of the crime. There are 214 of instances from 6 classes. The 9 features include glass material information, such as sodium, silicon, or iron. 5-fold cross validation is conducted on the entire dataset and the best score is reported.

Statlog Satellite Images (*satimage*). There are 4435 training examples and 2000 testing examples in this dataset. Each instance consists of the pixel values in the 4 spectral bands of each of the 9 pixels in the 3×3 neighborhood, that are contained in a frame of a Landsat satellite image. The objective is to classify the central pixels into 6 attributes, such as red soil, or cotton crop.

Statlog Letters (*letter*). This dataset comprises a collection of 20,000 stimuli, consisting of 26 categories of capital English letters on different fonts. Each stimulus was converted into 16 primitive numerical attributes, such as statistical moments and edge counts. The first 15000 items are chosen to be training examples and remaining 5000 items to be testing examples.

We have compared the classification accuracies achieved with the proposed sensitivity vector RBF, support vector machine [11], conventional RBF [2], and decision tree C4.5 [20]. The training examples and testing examples for all the methods are the same as those chosen for our proposed SenV-RBF. The experimental results show that, data classification based on the proposed sensitivity-based RBF classifier performs better than the conventional RBF and the C4.5. The exception to this, *glass*, indicates that from case to case the Gaussian kernel function may have some blind spots. It can also be seen that both the sensitivity-based RBF and the SVM achieve basically the same level of accuracy, but the former enjoys the advantage of being easy to control and suitable for multiple-class problems, as well as being robust against noisy data. The advantages are accrued from the fact that the sensitivity-based RBF makes an effective trade-off between structural risk minimization and empirical risk minimization. The sensitivity-based RBF is time-consuming in training, but this is not a problem, as it provides a very fast and accurate classification in run-time, our area of concern.

5 Conclusion and Future Work

The conventional approach to constructing an RBF network is to search for the optimal cluster centers among the training examples. This paper proposes a novel approach to RBF construction that uses critical vectors selected by sensitivity analysis. Sensitivity is defined as the expectation of the square of output deviation caused by the perturbation of RBF centers. In training, orthogonal least square incorporated with a sensitivity measure is employed to search for the optimal critical vectors. In classification, the selected critical vectors will take on the role of the RBF centers. Our experimental results show that our proposed RBF classifier performs better than conventional RBFs and C4.5. The sensitivity-based RBF can achieve the same level of accuracy as SVM, but strikes a balance between structural risk minimization and empirical risk minimization. The advantage of the sensitivity-based RBF is that it is suitable for large scale multi-class applications, since it is easy to control as well as robust against noisy data.

In some cases, support vectors are exactly the same as sensitivity-oriented critical vectors, but the relationship between these two kinds of vectors remains unknown. Our future work will include investigating the relationship between sensitivity-based RBF and SVM in classification.

References

1. M. Bianchini, P. Frasconi and M. Gori, Learning without local minima in radial basis function networks, *IEEE Transactions on Neural Networks*, **6(3):**749-756, 1995.
2. C. M. Bishop, Improving the generalization properties of radial basis function neural networks, *Neural Computation*, **3(4):**579-581, 1991.
3. C. M. Bishop, Training with noise is equivalent to Tikhonov regularization, *Neural Computation*, **7(1):**108-116, 1995.
4. C. L. Blake, and C. J. Merz, UCI Repository of machine learning databases. *School of Information and Computer Science, University of California, Irvine, CA.* [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html, 1998
5. S. Chen, C. F. Cowan and P. M. Grant, Orthogonal least squares learning algorithms for radial basis function networks, *IEEE Transactions on Neural Networks*, **2(2):**302-309, 1991.
6. S. Chen, E. S. Chng and K. Alkadhimi, Regularized orthogonal least squares algorithm for constructing radial basis function networks. *International Journal of Control*, **64(5):**829-837, 1996.
7. S. Chen, Y. Wu and B. L. Luk, Combined genetic algorithm optimization and regularized orthogonal least squares learning for radial basis function networks, *IEEE Transactions on Neural Networks*, **10:**1239-1243, 1999.
8. J. Y. Choi and C. H. Choi, Sensitivity analysis of multilayer perceptron with differentiable activation functions, *IEEE Transactions on Neural Networks*, **3(1):**101-107, 1992, 1992.
9. V. Kadirkamanathan and M. Niranjan, A function estimation approach to sequential learning with neural networks, *Neural Computation*, **5(6):**954-975, 1993.
10. Y. S. Hwang and S. Y. Bang, An efficient method to construct a radial basis function neural network classifier, *Neural Networks*, **10(8):**1495-1503, 1997.
11. W. Hsu and C. J. Lin, A comparison of methods for multi-class support vector machines, *IEEE Transactions on Neural Networks*, **13(2):**415-425, 2002.
12. K. Z. Mao, RBF neural network center selection based on fisher ratio class separability measure, *IEEE Transactions on Neural Networks*, **13(5):**1211-1217, 2002.
13. D. Michie, D. J. Spiegelhalter and C. C. Taylor, Machine learning, neural and statistical classification. [Online]. Available: http://www.liacc.up.pt/ML/statlog/datasets.html, 1994.
14. J. Moody and C. J. Darken, Fast learning in networks of locally-tuned processing units, *Neural Computation*, **1:**281-294, 1989.

15. M. J. L. Orr, Regularization in the selection of radial basis function centers, *Neural Computation*, **7:**606-623, 1995.
16. J. Park, and I. W. Sandberg, Approximation and radial basis function networks, *Neural Computation*, **5:**305-316, 1993.
17. C. Panchapakesan, M. Palaniswami, D. Ralph and C. Manzie, Effects of moving the centers in an RBF network, *IEEE Transactions on Neural Networks*, **13(6):**1299-1307, 2002.
18. S. W. Piche, The selction of Weight Accuracies for Madalines, *IEEE Transactions on Neural Networks,* **6(2):**432-445, 1995.
19. T. Poggio and F. Girosi, Networks for approximation and learning. *Proceedings of the IEEE*, **78:**1481-1497, 1990.
20. J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.
21. B. Scholkopf, K. K. Sung, C. J. C. Burges, F. Girosi, P. Niyogi, T. Poggio and V. Vapnik, Comparing support vector machines with Gaussian kernels to radial basis function classifiers, *IEEE Transactions on Signal Processing*, **45(11):**2758-2765, 1997.
22. M. Stevenson, R. Winter and B. Widrow, Sensitivity of feedforward neural networks to weight errors, *IEEE Transactions on Neural Networks,* 1(1):71-80, 1990.
23. Z. Wang, and T. Zhu, An efficient learning algorithm for improving generalization performance of radial basis function neural networks, *Neural Networks*, **13(4):**545-553, 2000.
24. W. Wright, Bayesian approach to neural network modeling with input uncertainty. *IEEE Transactions on Neural Networks*, **10(6):**1261-1270, 1999.
25. D. S. Yeung and X. Sun, Using function approximation to analyze the sensitivity of MLP with antisymmetric squashing activation function, *IEEE Transactions on Neural Networks*, **13(1):**34-44, 2002.
26. X. Zeng and D. S. Yeung, Sensitivity analysis of multilayer perceptron to input and weight perturbation, *IEEE Transactions on Neural Network*, **12(6):**1358-1366, 2001.
27. X. Zeng and D. S. Yeung, A quantified sensitivity measure for multilayer perceptron to input perturbation, *Neural Computation*, **15:**183-212, 2003.
28. J. M. Zurada, A. Malinowski and S. Usui, Perturbation method for deleting redundant inputs of perceptron networks. *Neurocomputing*, **14(2):**177-193, 1997.

A Chaotic Neural Network for the Maximum Clique Problem

Shenshen Gu and Songnian Yu

School of Computer Engineering and Science, Shanghai University,
149 Yan Chang Road,
Shanghai, P.R.China, 200072
gushenshen@163.com

Abstract. This paper applies a chaotic neural network (CNN) to solve the maximum clique problem (MCP), a classic NP-hard and computationally intractable graph optimization problem, which has many real-world applications. From analyzing the chaotic states of its neuron output and computational energy, we reach the conclusion that, unlike the conventional Hopfield neural networks (HNN) for the MCP such as steepest descent (SD) algorithm and continuous Hopfield dynamics (CHD) algorithm based on the discrete Hopfield neural network and the continuous Hopfield neural network respectively, CNN can avoid getting stuck in local minima and thus yields excellent solutions. Detailed analysis of the optimality, efficiency, robustness and scalability verifies that CNN provides a more effective and efficient approach than conventional Hopfield neural networks to solve the MCP.

Keywords: Maximum clique; Hopfield neural network; Chaotic neural network; Transiently chaotic neural network; Chaos

1 Introduction

The maximum clique problem is the problem of finding the largest clique in a given graph G, i.e. the maximum subset of vertices of G such that every pair of vertices are connected by an edge. The MCP is widely applied in various fields such as information retrieval, fault tolerance, classification theory, circuit design [1] and computer vision. In addition, some classic graph problems, such as the maximum independent set problem, can be transformed into the MCP. Nevertheless, this problem is proved to be NP-hard [2] and computationally intractable even to approximate well [3].

To approximate the maximum clique effectively and efficiently, many algorithms have been proposed such as local search algorithms [4], greedy algorithms [5], genetic algorithms [6] and so on. Among these algorithms, the Hopfield neural network-based algorithms are very promising for the MCP. Introduced by Hopfield and Tank in 1985, the HNN is widely utilized to solve a large number of classic combinatorial optimization problems such as the traveling salesman problem (TSP) [7]. As far as the MCP is concerned, Takefunji et al. [8] and

Jagota [9] have proposed several algorithms based on the HNN. All these algorithms have proved to be effective to some extent.

However, the HNN has some obvious defects. Precise values of parameters of the HNN are required in order to acquire feasible solutions [10], which worsens the robustness of the algorithm. Moreover, the HNN is apt to get stuck in local minima owing to the steepest descent dynamics of the method, which degrades its effectiveness.

Fortunately, by probing into the chaotic behavior of nonlinear dynamics, several chaotic neural networks have been proposed which can overcome the above two disadvantages. Among these are the chaotic simulated annealing neural network model with transient chaos by Chen and Aihara [11], chaotic simulated annealing neural network with decaying time step by Wang and Smith [12] and the Hopfield neural network with external noises by Asai. et al [13].

Up to now, these chaotic neural networks have been applied to solve the TSP [11] and the N-queen problem [14], and the experimental results reveal that the performance of these chaotic neural networks is considerably better than that of those conventional Hopfield neural networks.

In this paper, we propose a new algorithm based on a chaotic neural network, the transiently chaotic neural network (TCNN) by Chen and Aihara (1995), to solve the MCP. From analyzing the chaotic states of its neuron output and computational energy, it is justified that this model possesses rich chaotic behavior in solving the MCP, which prevents it from getting stuck in local minima. Detailed analysis of the optimality, efficiency, robustness and scalability also verifies that this model is more effective and efficient than other algorithms based on the conventional HNNs in solving the MCP.

This paper is organized as follows. In the next section, some fundamental introduction to the MCP and HNN is presented. In Section 3, two well-known algorithms based on HNN model, SD and CHD, are given. In Section 4, the defects of the HNN for the MCP are uncovered followed by the introduction to the TCNN for the MCP. In Section 5, the optimality, efficiency, robustness and scalability of the new algorithm are analyzed in detail. Finally, the conclusion of this paper is presented in Section 6.

2 Preliminaries

In this section, we start with the definitions of clique, maximal clique and maximum clique, and then introduce the Hopfield neural network briefly.

Definition 1. *Clique is a set of vertices in an undirected graph such that every two vertices are connected by an edge.*

Definition 2. *Maximal Clique is a clique that is not a strict subset of any other clique in a given graph.*

Definition 3. *Maximum Clique is the clique that has more vertices than any other clique in a given graph.*

From above definitions, it is obvious that the maximum clique must be a maximal clique, however a maximal clique may not be the maximum clique.

In Fig.1, the vertex sets: {a,b,c,e}, {a,b}, {a,b,e}, {b,c,d,e} are not clique, non-maximal clique, maximal-but-not-maximum clique and maximum clique respectively.

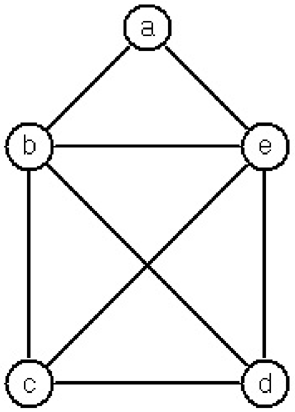


Fig. 1. Graph with maximal cliques {a,b,e},{b,c,d,e}

The maximum clique problem is the optimization problem of finding the maximum clique in a given graph. As is described in the previous section, the MCP has various real-world applications in the fields of computer science [15], VLSI design, economics [16], etc. In addition, many graph problems are directly related with it such as the maximum independent set problem.

In the next section, we introduce two approximation algorithms for the MCP based on the discrete HNN model and the continuous HNN model respectively, therefore we give brief description to these two HNN models as follows.

The Hopfield neural network is a recurrent network with N units connected with a symmetric matrix $W \equiv (\omega_{ij})_{N\times N}$, in which $\omega_{ii} = 0, (i = 1, 2, \ldots, N)$.

In the discrete HNN, the state of unit i is: $S_i \in \{0, 1\}$, and the input of unit i is: $n_i = \sum_{j=1}^{N} \omega_{ij} S_j + I_i$, where I_i is the input bias of unit i. The update strategy of these N units is to pick one unit and update its state at time $t+1$ as follows:

$$S_i(t+1) = \begin{cases} 1, & \text{if } n_i(t) > 0 \\ 0, & \text{if } n_i(t) < 0 \\ S_i(t), & \text{Otherwise} \end{cases} \tag{1}$$

which minimize the following energy function:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \omega_{ij} S_i S_j - \sum_{i=1}^{N} I_i S_i \tag{2}$$

For the continuous HNN, the state of unit i is: $S_i \in [0,1]$, and the update strategy of these N units is listed as follows:

$$S_i\left(t+1\right) = S_i\left(t\right) + \gamma\left(-S_i\left(t\right) + g_\lambda\left(WS_i\left(t\right) + \omega_0\right)\right), \left(i = 1, 2, \ldots, N\right) \qquad (3)$$

where $g\left(x\right) \equiv 1/\left(1 - e^{-\lambda x}\right)$ is a sigmoid and λ its gain, γ is the step size, which minimizes the following energy function:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\omega_{ij}S_iS_j - \sum_{i=1}^{N}I_iS_i + \sum_{i=1}^{N}\int_0^{S_i} g_\lambda^{-1}\left(S\right)dS \qquad (4)$$

3 Two HNN Algorithms for the MCP

In [8], Jagota proposed several algorithms based on the discrete HNN and the continuous HNN for the MCP, most of which were justified to be effective in terms of the average size of retrieved clique and the distribution of clique size. Here, the SD algorithm and the CHD algorithm based on the discrete HNNs and the continuous HNNs respectively are presented to give readers an overview of how HNNs are applied to solve the MCP.

For the SD algorithm, the state of unit i S_i, $(i = 1, , \ldots, N)$, which corresponds to vertex i in the given graph, is either 0 or 1, therefore the state of these N units is: $S = (S_1, S_2, \ldots, S_N) \in \{0,1\}^N$. In the SD algorithm, $W \equiv (\omega_{ij})_{N\times N}$ is a symmetric matrix, where $\omega_{ii} = 0, \omega_{ij} = 1$ if vertex i and j are connected and $\omega_{ij} = \rho, (\rho < 0)$ if vertex i and j are not connected. Switching S_i means changing the state of S_i, that is to say that if $S_i = 0$ currently, after switch it is replaced by 1, vice versa. The energy difference $\Delta E_i, (i = 1, 2, \ldots, N)$, is defined as follows:

$$\Delta E_i = \begin{cases} -\left(\sum\limits_{j=1}^{N}\omega_{ij}S_j + I_i\right), & \text{if } S\left(t-1\right) = 0 \\ \sum\limits_{j=1}^{N}\omega_{ij}S_j + I_i, & \text{if } S\left(t-1\right) = 1 \end{cases} \qquad (5)$$

Here we present the SD algorithm in forms of pseudocodes.

SD Algorithm
```
Initialize S= $\{0,1\}^N$ and W= $(\omega_{ij})_{N\times N}$
for $t \leftarrow 1$ to N
    for $i \leftarrow 1$ to N
        do calculate $\Delta E_i = \{\Delta E_1, \Delta E_2, \ldots, \Delta E_N\}$
            if $\Delta E_i = \min \Delta E$ and $\Delta E_i < 0$
                then switch unit i
```

Unlike the SD algorithm, in the CHD algorithm, the state of unit i S_i is a real value between 0 and 1, therefore the state of these N units is: $S = (S_1, S_2, \ldots, S_N) \in [0,1]^N$. Here, the CHD algorithm is listed as follows:

CHD Algorithm

```
Initialize S= $\{0.5 + Ramdom\,(-0.005, +0.005)\}^N$ and W= $(\omega_{ij})_{N\times N}$
$t \leftarrow 0$
do $S\,(t+1) = S\,(t) + \gamma\,(-S\,(t) + g_\lambda\,(WS\,(t) + \omega_0))$
    increase t by 1
until $S\,(t+1)$ is a fixed point
```

Jagota verified that, both in the SD algorithm and in the CHD algorithm, S converges to a set of fixed points. And the stable states of the SD and the CHD correspond to maximal cliques when $\rho = -4N$.

However, it is uncovered that the HNN may get stuck in local minima [9]. In the case of the MCP, the algorithms based on the HNN may become trapped into maximal cliques rather than the maximum clique, thus degrading its performance. In the next section, we verify that both SD and CHD may get stuck in maximal cliques, and we also probe into the direct causes and then introduce the TCNN for the MCP.

4 The Transiently Chaotic Neural Network for the MCP

Before introducing the TCNN for the MCP, let us first verify that both the SD algorithm and the CHD algorithm may get stuck in a maximal clique. To verify it, we just need to verify that the SD algorithm as well as the CHD algorithm will halt once they reach a maximal clique.

Theorem 1. *The SD algorithm will halt if it reaches a maximal clique.*

Proof. When SD reaches a maximal clique $V', (V' \subseteq V)$, in a given graph $G\,(V,E)$ at step $(t-1)$. Then we switch the state of S_k.

(1)Suppose that $S_k\,(t-1) = 1, (k = 1, 2, \ldots, N)$. After switching, $S_k\,(t) = 0$ and the energy difference $\varDelta E$ of SD is:

$$\varDelta E = E\,(t) - E\,(t-1) \tag{6}$$

$$\begin{aligned} &= -\tfrac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\omega_{ij}S_i\,(t)\,S_j\,(t) \;-\sum_{i=1}^{N}I_iS_i\,(t) \\ &-\left(-\tfrac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\omega_{ij}S_i\,(t-1)\,S_j\,(t-1) \;-\sum_{i=1}^{N}I_iS_i\,(t-1)\right) \end{aligned} \tag{7}$$

$$\begin{aligned} &= -\tfrac{1}{2}\sum_{j=1}^{N}\omega_{kj}S_k\,(t)\,S_j\,(t) \;- I_kS_k\,(t) \\ &-\left(-\tfrac{1}{2}\sum_{j=1}^{N}\omega_{kj}S_k\,(t-1)\,S_j\,(t-1) \;- I_kS_k\,(t-1)\right) \end{aligned} \tag{8}$$

$$= -\tfrac{1}{2}\left(\sum_{j=1}^{N}\omega_{kj}S_k(t)S_j(t) - \sum_{j=1}^{N}\omega_{kj}S_k(t-1)S_j(t-1)\right) \\ -I_k\left(S_k(t)-S_k(t-1)\right) \tag{9}$$

$$= \frac{1}{2}\sum_{j=1}^{N}\omega_{kj}S_j(t-1) + I_k \tag{10}$$

Because V' is a maximal clique and $S_k(t-1)=1$, we get that vertex $v_k \in V'$. Therefore, if $v_j \in V'$ then $S_j(t-1)=1$ and $\omega_{kj}=1$, else if $v_k \notin V'$ then $S_j(t-1)=0$ and $\omega_{kj}=\rho<0$

As a result $\Delta E>0$, i.e. S_k can not switch from 1 to 0 at step t.

(2)Suppose that $S_k(t-1)=0,(k=1,2,\ldots,N)$. After switching, $S_i(t)=1$ and the energy difference ΔE of SD is:

$$\Delta E = E(t) - E(t-1) \tag{11}$$

$$= -\tfrac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\omega_{ij}S_i(t)S_j(t) - \sum_{i=1}^{N}I_iS_i(t) \\ -\left(-\tfrac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\omega_{ij}S_i(t-1)S_j(t-1) - \sum_{i=1}^{N}I_iS_i(t-1)\right) \tag{12}$$

$$= -\tfrac{1}{2}\sum_{j=1}^{N}\omega_{kj}S_k(t)S_j(t) - I_kS_k(t) \\ -\left(-\tfrac{1}{2}\sum_{j=1}^{N}\omega_{kj}S_k(t-1)S_j(t-1) - I_kS_k(t-1)\right) \tag{13}$$

$$= -\tfrac{1}{2}\left(\sum_{j=1}^{N}\omega_{kj}S_k(t)S_j(t) - \sum_{j=1}^{N}\omega_{kj}S_k(t-1)S_j(t-1)\right) \\ -I_k\left(S_k(t)-S_k(t-1)\right) \tag{14}$$

$$= -\frac{1}{2}\sum_{j=1}^{N}\omega_{kj}S_j(t) - I_k \tag{15}$$

Because V' is a maximal clique, and $S_k(t-1)=0$, from definitions in Section 2, we get $V'\cup\{v_k\}$ is not a clique. Therefore, there exists a vertex $v_i \in V'$ such that v_i and v_k are not connected by an edge. Because $\omega_{ik}=\rho=-4N$ and $S_i(t)=1$, even if $\omega_{kj},(j=1,2,\ldots,i-1,i+1,\ldots,N)$, all equal to 1, $\sum_{j=1}^{N}\omega_{kj}S_j(t) \le -(4N+N-1) = -(3N-1)$. Taking $I_k \le 1$ into account, consequently $\Delta E>0$ and S_k can not switch from 0 to 1.

From the above proof it is obvious that the SD algorithm may get stuck in a maximal clique.

Theorem 2. *The CHD algorithm will halt if it reaches a maximal clique.*

It is verified that the local minima of CHD's energy function corresponds to a maximal clique. So we just need to verify that the energy function of the CHD strictly decreases at every step.

Proof. We first transform the CHD algorithm to the standard form of continuous Hopfield Neural Network as follows:

$$\begin{cases} n_i = \sum_{j=1}^{N} \omega_{ij} S_j + \omega_0 \\ \frac{dy}{dt} = -y_i + n \\ S_i = g_\lambda\left(y_i\right) \end{cases} \tag{16}$$

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \omega_{ij} S_i S_j - \sum_{i=1}^{N} I_i S_i + \sum_{i=1}^{N} \int_0^{S_i} g_\lambda^{-1}\left(S\right) dS \tag{17}$$

Because

$$\frac{dE}{dt} = \sum_{i=1}^{N} \frac{\partial E}{\partial S_i} \cdot \frac{dS_i}{dt} \tag{18}$$

and

$$\frac{\partial E}{\partial S_i} = -\sum_{j=1}^{N} \omega_{ij} S_j - I_i + g_\lambda^{-1}\left(S_i\right) \tag{19}$$

$$= -\sum_{j=1}^{N} \omega_{ij} S_j - I_i + y_i \tag{20}$$

$$= -\sum_{j=1}^{N} \omega_{ij} S_j - \omega_0 + y_i \tag{21}$$

$$= -\frac{dy_i}{dt} \tag{22}$$

Then, we get

$$\frac{dE}{dt} = \sum_{i=1}^{N} \left(-\frac{dy_i}{dt} \cdot \frac{dS_i}{dt}\right) \tag{23}$$

$$= \sum_{i=1}^{N} \left(-\frac{dy_i}{dS_i} \cdot \frac{dS_i}{dt} \cdot \frac{dS_i}{dt}\right) \tag{24}$$

$$= \sum_{i=1}^{N} \left(-\frac{dy_i}{dS_i} \cdot \left(\frac{dS_i}{dt}\right)^2\right) \tag{25}$$

Because $S_i = g_\lambda\left(y_i\right)$ strictly increases, therefore $y_i = g_\lambda^{-1}\left(S_i\right)$ also strictly increases.

As a result, $\frac{dE}{dt} < 0$.

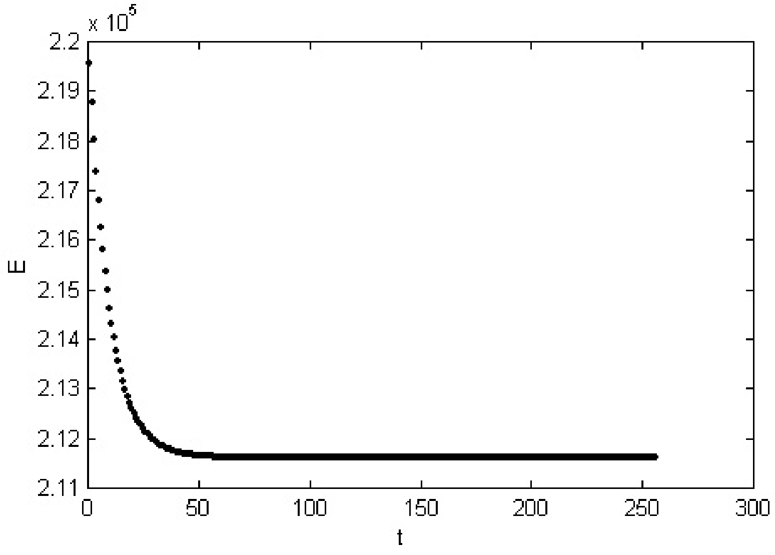


Fig. 2. Time evolution of the energy of the CHD approximating the maximum clique

From the above proof it is obvious that the CHD algorithm may get stuck in a maximal clique.

In Fig.2, the steepest descent dynamics of the CHD can be obviously observed.

Although the steepest descent dynamics of HNN guarantees the convergence and stabilization, it inevitably leads to two serious defects

(1)Acute values of parameters of the HNN are required in order to acquire feasible solutions.

(2)The HNN tends to become trapped in local minima.

Recall the graph in Fig.1, we apply the SD and the CHD to it. A round consists of 100 runs of these two algorithms, each with different set initial states. We find that 30 percent results of the SD and 25 percent results of the CHD are {a,b,c}, a maximal clique, rather than {b,c,d,e}, the maximum clique.

Therefore, we can draw the conclusion that the conventional HNNs for the MCP are apt to stuck in local minima, in other words, the conventional HNNs tend to reach maximal cliques rather than the maximum clique.

To overcome these two disadvantages of the HNN, some chaotic neural networks have been proposed by probing into the chaotic behavior of nonlinear dynamics. These chaotic neural networks have been applied to solve several combinatorial problems such as TSP and N-queen problem. And the experimental results reveal that the performance of these CNNs is considerably better than that of those conventional HNNs.

In this paper, we apply a kind of these chaotic neural networks, a transiently chaotic neural network by Chen and Aihara, to the MCP to see whether chaotic neural networks are also effective to solve the MCP. In this section, we describe the framework of TCNN for the MCP, and in the next section, we present detailed analysis of the optimality, efficiency, robustness and scalability of this algorithm.

The framework of TCNN is presented as follows:

$$\begin{cases} x_i(t) = 1/\left(1 + e^{-y_i(t)/\epsilon}\right) \\ y_i(t+1) = \kappa y_i(t) + \alpha \left(\sum\limits_{j=1, i \neq j}^{N} \omega_{ij} x_j(t) + I_i \right) - z_i(t)\left(x_i(t) - I_0\right) \\ z_i(t+1) = (1-\beta) z_i(t) \end{cases}$$

$(i = 1, 2, \ldots, N)$, where

$\omega_{ij} = \omega_{ji}$; $\omega_{ii} = 0$; $\sum\limits_{j=1, i \neq j}^{N} \omega_{ij} x_j(t) + I_i = -\frac{\partial E}{\partial x_i}$

E=energy function,

$x_i(t)$=output of neuron i,

$y_i(t)$=internal state of neuron i,

$z_i(t)$=self feedback connection weight ,$z_i(t) \geq 0$,

I_0=positive parameter,

I_i=input bias of neuron i,

α=positive scaling parameter for inputs,

β=damping factor of the time dependent $z_i(t)$, $(0 \leq \beta \leq 1)$

κ=damping factor of nerve membrane ,$(0 \leq \kappa \leq 1)$,

ϵ=steepness parameter of the output function ,$(\epsilon > 0)$.

To apply the TCNN to approximate the MCP, we adapt this model and get the algorithm as follows:

TCNN for the MCP____

```
Initialize $y_i(0) = \{0.5 + Ramdom(-0.005, +0.005)\}^N$ ,W and $z_i(0)$
$t \leftarrow 0$
    do
        for $i \leftarrow 1$ to N
            $x_i(t) = 1/\left(1 + e^{-y_i(t)/\varepsilon}\right)$
            $y_i(t+1) = ky_i(t) + \alpha \left( \sum_{j=1, i \neq j}^{N} \omega_{ij} x_j(t) + I_i \right) - z_i(t)(x_i(t) - I_0)$
            $z_i(t+1) = (1-\beta) z_i(t)$
        increase t by 1
    until E converges
```
____

Based on a large number of experiments, we find that the best results can be gained frequently if $\alpha \in [0, 0.6]$ and $z_0 \in [10, 20]$.

We also apply the TCNN to the graph in Fig.1. A round consists of 100 runs of the TCNN, each with a different set of initial states. We surprisingly find that all the results are {b,c,d,e}, the maximum clique. It seems that this algorithm is considerably more effective than the SD and the CHD.

It is obvious that the TCNN may be a promising method to solve the MCP. In the next section, we will analyze the optimality, efficiency, robustness and scalability of the TCNN in detail, and probe into the primary cause that prevents the TCNN from getting stuck in local minima.

5 Experimental Results and Analysis

In this section we apply the TCNN algorithm described in the previous section to solve the MCP. To begin with, the optimization performance of the TCNN is studied in terms of optimality, efficiency, robustness and scalability in a two-parameter space. In the case of MCP, optimality is defined as whether the maximum clique is obtained in a test; efficiency is defined as the number of iterations for convergence; robustness refers to the distribution of feasible regions; and scalability is concerned with the dependency of these measures on problem size. The experimental results are presented in Fig.3 in detail.

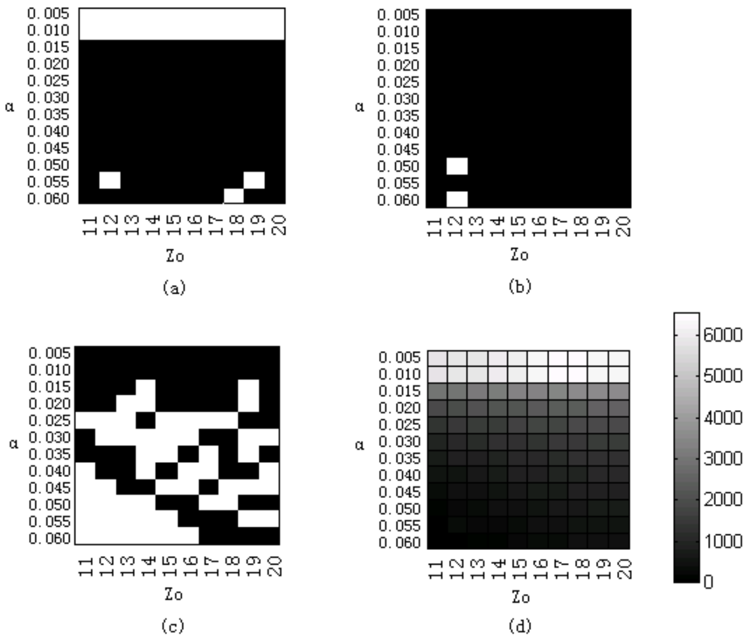


Fig. 3. Optimality and efficiency graphs of the TCNN approximating the maximum clique. (a),(b) and (c) shows optimality for |V|=28, 64 and 256 respectively. (d) shows the number of iterations for |V|=64.

The parameters are set as follows: $\rho = -4N, \omega_0 = \frac{|\rho|}{4}, \epsilon = 1, \beta = 0.001, \kappa = 0.9$. Theoretically, the parameters α and z_0 are the main parameters concerned with the balance of steepest descent and nonlinear feed back terms. Based on a large number of tests, we find that best results can be gained frequently if α is selected between 0 and 0.6 and z_0 is set between 10 and 20. Therefore, the parameter space is defined by α and z_0.

Fig.3(a)-(c) show the optimality obtained for |V|=28, 64 and 256 respectively. And Fig.3(d) shows the number of iterations for |V|=64. In Fig.3 (a)-(c), a black region means the maximum clique is obtained. It is obvious that, in Fig.3(a) and (b), the feasible region account for a considerably large portion of the parameter space, suggesting good robustness in choosing parameters when the number of vertices is medium. In addition, the feasible region does not shrink obviously when the size of problem increases, thus revealing good scalability. Although, presented in Fig.3(d), TCNN fails 55 in 120 cases in obtaining the best result, it is obvious that there are only 5 in 40 cases fail when the parameter α is selected between 0.005 and 0.020. This suggest us that the parameter α palys an important role in effecting the optimality, robustness and scalability.

In Fig.3(d), the grey-scale measures efficiency, darker shade means fewer iterations to reach convergence, in other words higher efficiency. Because both Fig.3(b) and (d) have the same problem size of |V|=64, we find that the TCNN can reach the maximum clique in a 64-vertics graph within 52 iterations when α and z_0 are set as 0.6 and 13 respectively. However, the iterations of the SD and the CHD to reach a maximal clique in this size of problem are both 64. This suggests that the TCNN has a high efficiency. From comparing Fig.3(b) and (d), we also find that the parameter α plays a more important role than the parameter z_0 does in effecting the optimality. And it can be obviously observed that with z_0 decreased, efficiency increased.

In Fig.4, which shows the time evolution of the computational energy E with |V|=64, $\alpha = 0.05$ and $z_0 = 15$, we can observe the overall neuron dynamics. The rich chaotic behavior is obvious when $t < 230$,and then the value of energy deduced gradually and finally reached a global minimum.

It suggests that, unlike the conventional HNN, the TCNN starts from deterministically chaotic dynamics, through a reversed period-doubling route with decreasing the value of z which corresponds to the temperature of usual annealing, finally reaches a stable equilibrium solution [10]. This distinctive characteristic of the TCNN may enhance its performance to solve the MCP.

Furthermore, we vary the values of the parameter α to examine its influence on the neural dynamics. Fig.5(a) and (b) are time evolutions of $S_i\left(t\right)$ with $\alpha = 0.05$ and $\alpha = 0.6$ respectively. In Fig.5(a) the chaotic dynamics of $S_i\left(t\right)$ lasts a long period, on the other hand, in Fig.5(b), the chaotic dynamics of $S_i\left(t\right)$ vanishes quickly. Therefore, it is obvious that the parameter α partially effect the bifurcation speed of the chaos. And in Fig.3(a), we can clearly observed that no maximum clique is obtained when α is set as 0.05 and 0.10. This may suggest that a extremely long period of chaos may defect the performance of this algorithm in some cases.

The performances of the TCNN have been experimented on graphs selected among those collected in the benchmark of the second DIMACS challenge. The results obtained by the TCNN on DIMACS graphs are listed in Table 5.1. The

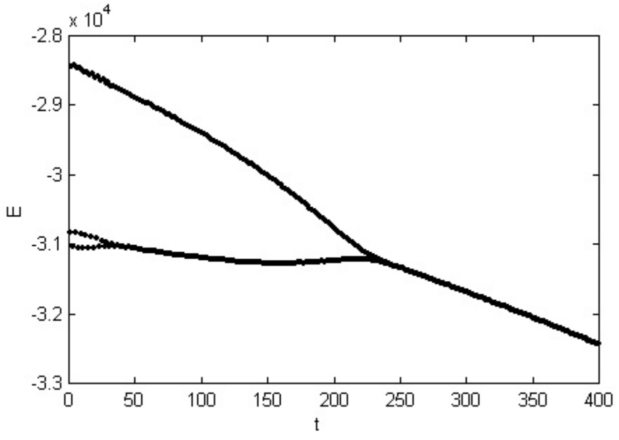


Fig. 4. Time evolution of the Energy of the TCNN approximating the maximum clique

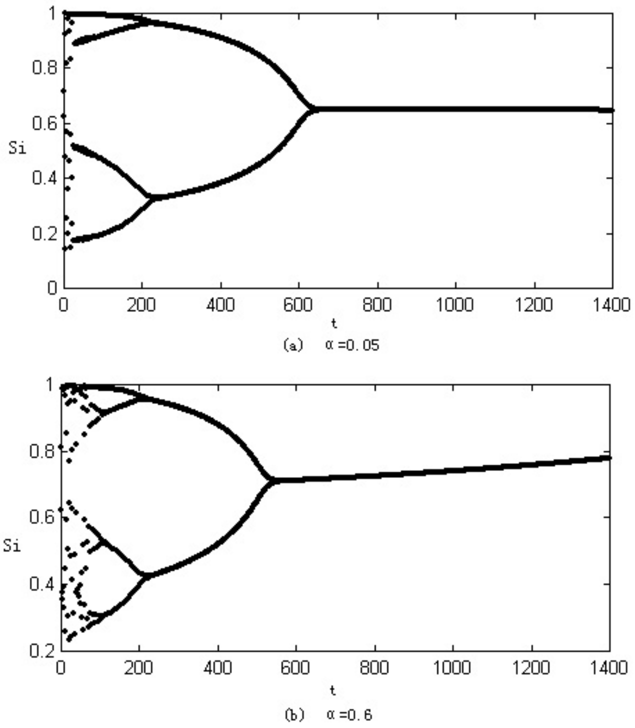


Fig. 5. Time evolutions of the single neuron model with different values of α (a)$\alpha = 0.05$;(b)$\alpha = 0.6$

first column (Graph) contains the name of the instances; the second (|V|) contains the number of vertices; the third (TCNN) reports the size of the cliques approximated by the TCNN which are given in bold if they coincide with the best results in the forth column, and the forth (Best Result) gives the best results obtained by other algorithms up to now.

Table 1 shows that the solutions found by the TCNN is considerably good. On 23 instances out of the 29 of the DIMACS challenge, the TCNN find a result equals to the best.

Table 1. Results on the DIMACS benchmark instances

Graph(V,E)	|V|	TCNN	Best Result
Hamming6-2	64	**32**	32*
Hamming6-4	64	**4**	4*
Hamming8-2	256	**128**	128*
Hamming8-4	256	**16**	16*
Hamming10-2	1024	**512**	512*
p.hat300-1	300	**8**	8*
p.hat300-2	300	**25**	25*
p.hat300-3	300	**36**	36*
p.hat500-1	500	**9**	9*
p.hat500-2	500	**36**	36*
p.hat500-3	500	47	50
p.hat700-1	700	10	11*
p.hat700-2	700	**44**	44*
p.hat700-3	700	59	62*
p.hat1000-1	1000	**10**	10
p.hat1000-2	1000	**46**	46
p.hat1000-3	1000	**68**	68
Johnson8-2-4	28	**4**	4*
Johnson8-4-4	70	**14**	14*
Johnson16-2-4	120	**8**	8*
c-fat200-1	200	**12**	12*
c-fat200-2	200	**24**	24*
c-fat200-5	200	**58**	58*
c-fat500-2	500	**26**	26*
c-fat500-5	500	**64**	64*
sanr200-0.7	200	17	18*
sanr200-0.9	200	41	42*
sanr400-0.5	400	12	13*
sanr400-0.7	400	**21**	21

6 Conclusion

In this paper, we adapt a new algorithm based on a chaotic neural network, called the transiently chaotic neural network, attempting to solve the maximum clique problem, a classic NP-hard and computationally intractable graph optimization problem, more effectively. Unlike those conventional Hopfield neural network-based algorithms which are verified in this paper may become trapped into maximal cliques, the algorithm proposed in this paper can avoid getting stuck in local minima thanks to its rich chaotic behavior in approximating the MCP. Detailed analysis of the optimality, efficiency, robustness and scalability as well as the results on DIMACS benchmark instances justify that this algorithm is more effective and efficient than conventional Hopfield neural networks for the MCP. However, we find that the parameter α has significant effect on the performance of this algorithm. Therefore, further developments and studies are still expected in the future.

References

1. S. Wimer, R.Y. Pinter and J. Feldman. Optimal chaining of CMOS transistors in a functional cell. IEEE Transactions on Computer-Aided Design, Vol:6, pp. 795-801, 1987.
2. R.M. Karp. Reducibility among combinatorial problems. Complexity of Computer Computations, pp. 85-103, 1972.
3. S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science, pp. 14-23, 1992.
4. R. Battiti and M. Protasi. Reactive local search for the maximum clique problem. Proceedings of the Workshop on Algorithm Engineering (WAE'97), pp. 74-82, 1997.
5. D.S. Johnson. Approximation algorithms for combinatorial problems. Journal of Computer. System, Science, Vol. 9, pp. 256-278, 1994.
6. T.N. Bui and P.H. Eppley. A hybrid genetic algorithm for the maximum clique problem. Proceeding of the 6th International Conference on Genetic Algorithms, pp. 478-484, 1995.
7. J.J. Hopfield and D.W. Tank. Neural computation of decisions in optimization problems. Biological Cybernetics, vol. 52, pp. 141-152, 1985.
8. N. Funabiki, Y. Takefuji, and K. Lee. A neural network model for finding a near-maximum clique. Parallel and Distributed Computing, vol. 14(3), pp. 340-344, 1992.
9. A. Jagota. Approximating maximum clique with a Hopfield network. IEEE Transactions on Neural Networks, vol. 6(33), pp. 724-735, 1995.
10. G.V. Wilson and G.S. Pawley. On the stability of the TSP algorithm of Hopfield and Tank. Biological Cybernetics, vol. 58, pp. 63-70, 1988.
11. L. Chen and K. Aihara. Chaotic simulated annealing by a neural network model with transient chaos. Neural Networks, vol. 8(6), pp. 915-930, 1995.
12. L. Wang and K. Smith. On chaotic simulated annealing. IEEE Transactions on Neural Networks, vol. 9(4), pp. 716-718, 1998.
13. H. Asai, K. Onodera, T. Kamio and H. Ninomiya. A study of Hopfield neural networks with external noises, Proceedings of the IEEE International Conference on Neural Networks, Perth, vol. 4, pp. 1584-1589, 1995.

14. T. Kwok and K.A. Smith. Experimental analysis of chaotic neural network models for combinatorial optimization under a unifying framework., Neural Networks, vol.13, pp.731-744, 2000.
15. N. Deo. Graph Theory with Applications to Engineering and Computer Science, Prentice-Hall, Englewood cliffs, NJ, 1974.
16. G. Avondo-Bodeno. Economic Applications of the Theory of Graphs, Gordon and Breach Science Publishers, New York, 1962.

Wavelet Network with OLS Optimization for Speech Signal Processing

Fei Chen, Daming Shi, and Geok See Ng

School of Computer Engineering, Nanyang Technological University,
Singapore 639798
chenfei@pmail.ntu.edu.sg, {asdmshi, asgsng}@ntu.edu.sg

Abstract. Taking advantage of both the scaling property of wavelets and the high learning ability of neural networks, wavelet networks have recently emerged as a powerful tool for many applications in the field of signal processing, such as data compression and function approximation. It is an implementation of wavelet transform, decomposing a signal into a series of scaled and translated wavelets. In the construction of wavelet network, the architecture must be pruned to reduce the system complexity, as well as to increase the generalization capability. Thus, the Orthogonal Least Square (OLS) algorithm is incorporated into the wavelet network to achieve the goal. Our experiments show that the OLS algorithm achieves the best results among all the network pruning algorithms, and the optimal wavelet network outperforms other traditional speech synthesis methods.

1 Introduction

Speech signal processing is an emerging technology with a wide range of potential applications, in which many methods from other areas are involved [1]. The examples are function approximation, statistical inference and machine learning.

In speech signal processing, especially in speech analysis and synthesis area, Pitch Synchronous Overlap Add (PSOLA) [2] and sinusoidal model [3] play an important role in controlling and adjusting the waveform of signals. PSOLA analyzes signals by window technique, dividing an original speech signal into a sequence of separate and often overlapping short-term signals. The idea of sinusoidal model is to decompose a signal into a set of sinusoids with time-varying amplitudes and frequencies. Both techniques are Fourier analysis based methods, and have achieved rather success in speech signal processing field.

However, the speech signal is non-stationary signal, in which there are many abrupt changes, such as at the beginning of a pitch (Glottal Closure Instant). Analyzing the speech signal with windows or sinusoids is not well suited. When comparing with the Fourier analysis, wavelet analysis processes signals at different scales [4]-[7]. Therefore, we can notice the gross features at large scales, and detailed changes at small scales. Additionally, since the wavelet is a local function, it is good at approximating any signals in finite domains.

Neural networks provide a linear algorithm capable of representing complex non-linear mapping, and can approximate any regular input sequence [8]. With the prosperity of neural network in the 1980s, many machine learning techniques were applied to signal processing. The neural network's learning ability from training data makes it a useful tool for signal processing. There are many signal processing applications employing neural network models [9, 10].

Inspired by the similarity between discrete wavelet transform and neural networks, Zhang and Benveniste [11] proposed wavelet network in 1992. They constructed a special feed-forward neural network supported by the wavelet theory. In the wavelet network, after constructing a wavelet library, which consists of a set of scaled and translated wavelets (regressors), some network pruning algorithms were employed to select the most significant regressors [12].

OLS algorithm [13] is to find the regressors, which provide the most significant contribution to approximation error reduction. It has been widely accepted as an effective method for regressor selection in RBF network [14]-[16]. Thus we will first incorporate OLS algorithm into the wavelet network, and then exploit its potential in speech signal processing.

The rest of the paper is organized as follows. We first briefly review the structure of wavelet network and introduce the modified OLS algorithm for network pruning in Section 2. In Section 3, experiments are conducted to demonstrate the effectiveness of the optimized wavelet network, and point out its advantages and disadvantages from the speech signal processing view. Finally, we conclude the features of wavelet network and outline our future work in the last section.

2 Wavelet Network

Given a classification or regression problem, the architecture of a wavelet network is exactly specified by how many particular wavelets required. In this research, a library of wavelets will be selected to be the candidate hidden neurons (wavelets), and then the optimal architecture is constructed by pruning hidden neurons.

2.1 Network Architecture

As an advanced alternative over classical Fourier analysis, wavelets have been successfully used in many aspects of signal processing, such as de-noising, compressing, edge detection, and so on. The fundamental idea behind wavelets is to process data at different scales or resolutions. In such a way, wavelets provide a time-scale presentation of a sequence of input signal.

Based on the theory of discrete wavelet transform, Zhang and Benveniste [11] tried to link the network coefficients with this appropriate transform. In 1992, they proposed wavelet network for approximation, which had an explicit link between the network coefficients and the discrete wavelet transform. In this way, they took advantage of the concept of discrete wavelet transform to initialize the coefficients. The wavelet network structure is of the following form

$$f(x) = \sum_{i=1}^{N} w_i \psi(diag(s_i)(x - t_i)) + c^T x + \overline{g} \tag{1}$$

where N is the number of wavelets,

w_i is the weight of the ith wavelet,

$\psi(x)$ is the mother wavelet,

t_i is the translation factor of the ith wavelet,

s_i is the dilation factor of the ith wavelet,

c is a weight matrix from the input to the output,

$\overline{g}$ is a bias to the output.

2.2 Network Initialization

A wavelet library, W, which consists of discretely dilated and translated versions of a given wavelet ψ, should be constructed according to the training data set. The wavelet family is

$$\left\{\psi\left(s_0^n x - mt_0\right) : n \in S_s, m \in S_t(n)\right\}. \tag{2}$$

In equation (2), s_0, $t_0 > 0$ are two scalar constants defining the discretization step sizes for dilation and translation. s_0 is typically dyadic. S_t is a finite set related to the input data domain D. S_s is also a finite set with the largest scale corresponding to the size of D.

The maximum scale level of the wavelet network should be determined first, and consequentially all the possible translation values will be calculated at each scale level.

2.3 Network Pruning Algorithm

After the construction of wavelet library, the number of wavelets, N_s, may be decided manually or automatically. Therefore, in the N-candidate wavelet library, the goal is to find the best N_s wavelet regressors, which minimize the error between output vectors and expected vectors.

This is a classical topic in the hidden neuron construction of RBF networks. There are two different ways to specify the number of hidden neurons. One way, explicated by Bishop [8], is to cluster the training examples and then assign a neuron to each cluster. The other way, after Piggio [17], is to learn mapping from a set of examples, that is, to create a neuron for each training example and then to prune the hidden neurons by example selection.

Orthogonal least square algorithm [13] has been widely used in pruning RBF networks. It combines the orthogonal transform with the forward regression procedure to select regressors from a large candidate regressor set. The advantage of employing OLS is that the responses of the hidden layer neurons are decorrelated so that the contribution of individual candidate neurons to the approximation error reduction can be evaluated independently.

In the forward orthogonalization procedure, only one column of regressor matrix is orthogonalized at the K th iteration. The K th column is made orthogonal to each of the $K-1$ previously orthogonalized columns and the operation for $K=2,\ldots,N$ is repeated.

Meanwhile, the optimal number of wavelets may be determined by introducing a criterion, Akaike's final prediction error criterion (FPE) [18]. It is actually the mean square error (MSE) of the function $f(x)$, multiplied by a factor of penalty in terms of the number of parameters. Its definition is given by:

$$FPE(f)=\frac{1+n_p/N_t}{1-n_p/N_t}\frac{1}{2N_t}\sum_{n=1}^{N_t}(f(x_n)-y_n)^2 \tag{3}$$

where n_p is the number of parameters in the estimator,

(x_n,y_n) are training data, and

N_t is the length of training data.

When the FPE value begins to increase monotonically, if the delta MSE is also less than a threshold value, ε, the pruning procedure will be stopped. The stopping criteria are given by

$$\begin{cases} FPE^{(k)}-FPE^{(k-1)}\geq 0 \\ \Delta MSE<\varepsilon \end{cases} \tag{4}$$

The wavelets corresponding to the first K columns will be selected to construct the wavelet network.

2.4 Network Training

The network training is an iterative procedure based on error back propagation. Let $\Theta=(\overline{g},c,w_i,t_i,s_i)$, then the objective is to minimize the expectation of the following function,

$$m\left(\Theta,x^{(k)},y^{(k)}\right)=\frac{1}{2}[f_\Theta\left(x^{(k)}\right)-y^{(k)}]^2 \tag{5}$$

Thus, a stochastic gradient algorithm is used to recursively adjust these parameters as follows,

$$\Theta^{(k)} = \Theta^{(k-1)} - \gamma^{(k)} grad\left(m\left(\Theta^{(k-1)}, x^{(k)}, y^{(k)}\right)\right) \tag{6}$$

where $\gamma^{(k)}$ means the adaptive learning rate of the *k*th step.

3 Experimental Results

In this section, we will show the performance of wavelet network on a segment of speech signal. The sampling rate of the speech signal is 22.50 kHz, and the bits per sample are 16. The test segment has 1000 samples. The mother wavelet function is Mexican Hat wavelet, defined by

$$\psi(x) = \sqrt{2}\left(d - \|x\|^2\right) e^{-\frac{\|x\|^2}{2}} \tag{7}$$

where $\|x\|$ is the Euclidean norm of x and d is the dimension of input data.

3.1 Comparison among Network Pruning Algorithms

In the following, we briefly introduce three different network pruning algorithms used in [12].

Algorithm 1: Residual based selection. The idea of this method is to select, for the first stage, the wavelet in W that best fits the training data, and then repeatedly select the wavelet that best fits the residual of the fitting of previous stage.

Algorithm 2: Stepwise selection by orthogonalization. This method is similar to OLS algorithm. It first selects the wavelet in the W that best fits the training data, and then repeatedly selects the wavelet that best fit the training data while working together with the previously selected wavelets.

Algorithm 3: Backward elimination. This method first builds the regression with all the wavelets in W, and then eliminates one wavelet in each step while trying to increase as less as possible the residual.

Generally, to these network pruning algorithms, the stopping criterion $FPE^{(k)} - FPE^{(k-1)} \geq 0$ itself is enough to find an appropriate number of wavelets. However there are some cases that $\Delta MSE < \varepsilon$ provides a more recommendable number of wavelets to the wavelet network. The following figure is an example of these cases.

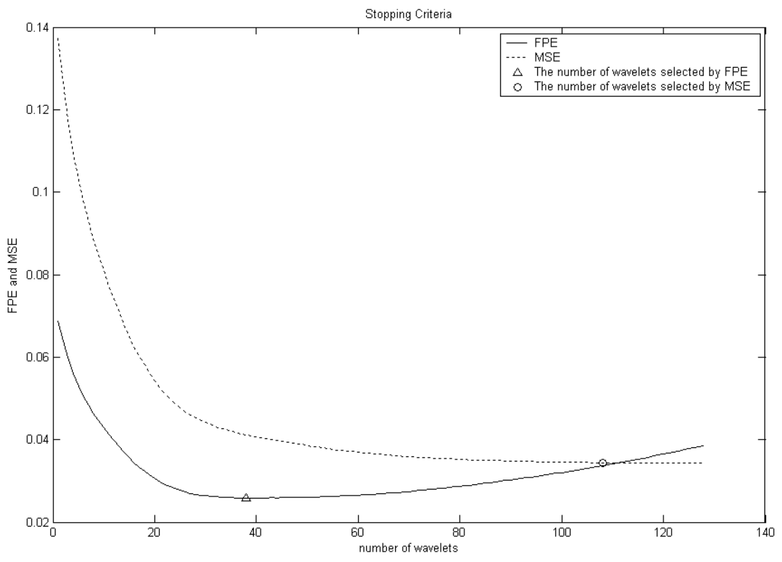


Fig. 1. Stopping criteria. In most cases, the FPE criterion can provide the optimal number of wavelets to the wavelet network. However, in this figure, the number of wavelets selected by MSE is more recommendable.

On our 1000 samples speech signal, we have compared the approximation performance between the three network pruning algorithms and the OLS from scale level 4 to level 9.

Table 1. Mean Square Error and Final Prediction Error of the approximation results of wavelet networks with different pruning algorithms

Scale Level	Algorithm 1		Algorithm 2		Algorithm 3		OLS	
	MSE	FPE	MSE	FPE	MSE	FPE	MSE	FPE
4	0.11059	0.05551	0.09383	0.04767	0.09383	0.04767	0.09083	0.04624
5	0.07510	0.03808	0.04639	0.02419	0.05195	0.02704	0.04627	0.02408
6	0.02446	0.01273	0.01153	0.00646	0.01260	0.00703	0.01197	0.00668
7	0.00558	0.00300	0.00280	0.00177	0.00282	0.00176	0.00282	0.00179
8	0.00249	0.00143	0.00070	0.00050	0.00100	0.00068	0.00073	0.00052
9	0.00158	0.00095	0.00011	0.00010	0.00013	0.00011	0.00005	0.00005

It can be seen that the OLS performs better than other network pruning algorithms. Those methods, directly adding or eliminating regressors, do not take account of the correlation between regressors. In contrast, the OLS algorithm decouples the correlations among the responses of candidate wavelets. With correlation analysis, the individual contribution of each wavelet to the approximation error reduction can be calculated. That is why the OLS algorithm can achieve better results.

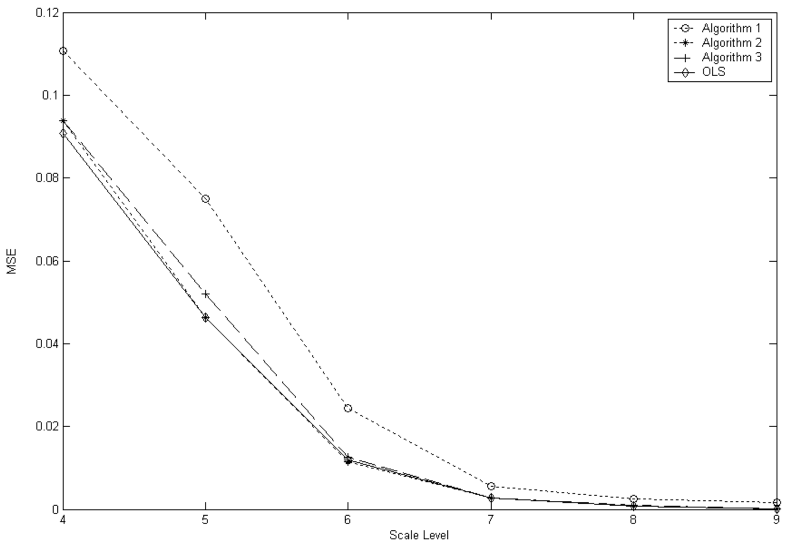


Fig. 2. Mean Square Error of the approximation results of wavelet networks with different pruning algorithms

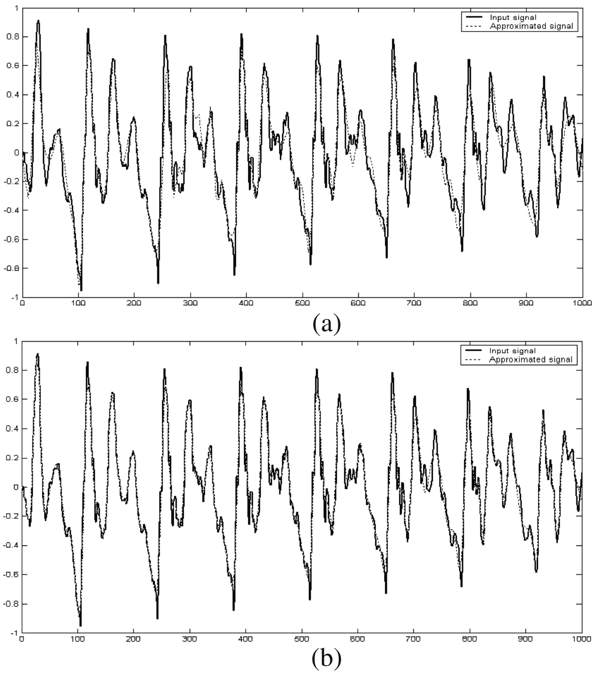


Fig. 3. Speech signal approximation by wavelet network. (a) Approximated signal after network pruning. (b) Approximated signal after training.

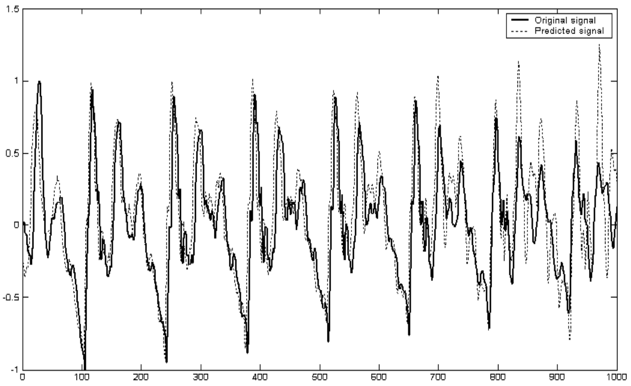


Fig. 4. Predicted signal by wavelet network

Table 2. Performance comparison between wavelet network and other existing speech models

	MSE	FPE
Wavelet Network	0.00349	0.00257
PSOLA	0.00728	0.00369
Sinusoidal Model	0.01375	0.00817

3.2 Comparison between Wavelet Network and Other Speech Models

In this section, we will evaluate the analysis / re-synthesis ability of wavelet network by comparing it with some existing speech processing models.

The approximation result of wavelet network on the segment of speech signal is shown in Fig. 3. The mean square error after initialization is 0.01511 (Fig. 3(a)), and is reduced to 0.00349 after 20 iterations training (Fig. 3(b)).

We can see, with the network pruning algorithm, the information of input data is effectively used to achieve a good initial state. Also due to the effective initialization, the training iteration is greatly reduced compared with common neural network models. The wavelet network reaches a relatively stable state rapidly.

To investigate the prediction capability of wavelet network, we further employed the previous n samples as input vector to predict the $n+1$ sample. In the experiment, we selected $n = 140$, and used the first 400 samples as the training data. Therefore, the input data is a 140×400 matrix. After training, the predicted signal is shown in Fig. 4, which quite matches the original signal.

We are now in a position to compare the approximation and prediction capability among the wavelet networks and some other representative methods on speech signal processing. The performance comparison among wavelet network, PSOLA and sinusoidal model is shown in Table 2. Here, the MSE is chosen to assess the approximation capabilities, whereas FPE is used to evaluate the prediction capabilities.

From the above table, we can see that wavelet network outperforms both PSOLA and sinusoidal model. The advantages of wavelet network include signal analysis at different scales, as well as high learning ability. Furthermore, with its flexible structure, it can achieve different accurate results by simply adjusting the number of wavelets. Since sinusoidal model extracts tracks in frequency domain, its waveform approximation result in time domain is lower than both PSOLA and wavelet network.

4 Conclusions and Future Work

In this paper, wavelet network has been analyzed and successfully applied to speech signal processing. First, we have employed the OLS algorithm to optimize the structure of wavelet network. Then, we have utilized the wavelet network in speech signal processing. The experimental results have shown the efficiency of OLS algorithm in network pruning. Our experiments have also shown that the optimized wavelet network has higher approximation and prediction ability than PSOLA and sinusoidal model.

Our future work includes the construction of the optimal wavelet network so as to improve its approximation and prediction abilities. The currently used OLS selects wavelets based on correlation analysis. The idea constructing RBF network based on support vectors in [19, 20] is worth borrowing.

References

1. Damper, R.I.: Data-Driven Techniques in Speech Synthesis. Kluwer Academic Publishers. (2001)
2. Moulines, E., Charpentier, F.: Pitch Synchronous Waveform Processing Techniques For Text-To-Speech Synthesis Using Diphones. Speech Communication, Vol. 9. (1990) 453-467
3. McAulay, R.J., Quatieri, T.F.: Speech Analysis / Synthesis Based On A Sinusoidal Representation. IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-34. (1986) 744-754
4. Mallat, S.G.: A Theory of Multiresolution Signal Decomposition: The wavelet representation. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 11. (1989) 674-693
5. Daubechies, I.: The Wavelet Transform, Time-Frequency Localization and Signal Analysis. IEEE Transactions on Information Theory, Vol. 36. (1990) 961-1005
6. Mallat, S.G., Zhong, S.: Characterization of Signals from Multiscale Edges. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 14, No. 7. (1992) 710-732
7. Resnikoff, H. L. Wells, R. O.: Wavelet Analysis: The Scalable Structure of Information. Springer Verlag (1998)
8. Bishop, C.M.: Improving the Generalization Properties of Radial Basis Function Neural Networks. Neural Computation, Vol. 3, No. 4. (1991) 579-588
9. Cocchi, G., Uncini, A.: Subband Neural Networks Prediction for On-Line Audio Signal Recovery. IEEE Transactions on Neural Networks, Vol. 13, No. 4. (2002) 867-876
10. Dorronsoro, J.R., López, V., Cruz, C.S., Sigüenza, J.A.: Autoassociative Neural Networks and Noise Filtering. IEEE Transactions on Signal Processing, Vol. 51, No. 5. (2003) 1431-1438

11. Zhang, Q., Benveniste, A.: Wavelet Network. IEEE Transactions on Neural Networks, Vol. 3, No. 6. (1992) 889-898
12. Zhang, Q.: Using Wavelet Network in Nonparametric Estimation. IEEE Transactions on Neural Networks, Vol. 8, No. 2. (1997) 227-236
13. Chen, S., Cowan, C.F., Grant, P.M.: Orthogonal Least Squares Learning Algorithms for Radial Basis Function Networks. IEEE Transactions on Neural Networks, Vol. 2, No. 2. (1991) 302-309
14. Chen, S., Chng, E.S., Alkadhimi, K.: Regularized Orthogonal Least Squares Algorithm for Constructing Radial Basis Function Networks, International Journal of Control, Vol. 64, No. 5. (1996) 829-837
15. Chen, S., Wu, Y., Luk, B.L.: Combined Genetic Algorithm Optimisation and Regularised Orthogonal Least Squares Learning for Radial Basis Function Networks. IEEE Transactions on Neural Networks, Vol.10, No.5. (1999) 1239-1243
16. Gomm, J.B., Yu, D.L.: Selecting Radial Basis Function Network Centers with Recursive Orthogonal Least Squares Training. IEEE Transactions on Neural Networks, Vol. 11. (2000) 306-314
17. Poggio, T., Girosi, F.: Network for Approximation and Learning. Proceedings of the IEEE, Vol. 78, No. 9. (1990) 1481-1497
18. Akaike, H.: Fitting Autoregressive Models for Prediction. Annals of the Institute of Statistical Mathematics, Vol. 21. (1969) 243-347
19. Scholkopf, B., Sung, K.K., Burges, C.J.C., Girosi, F., Niyogi, P., Poggio, T., Vapnik, V.: Comparing Support Vector Machines with Gaussian Kernels to Radial Basis Function Classifiers. IEEE Transactions on Signal Processing, Vol. 45, No. 11. (1997) 2758-2765
20. Vapnik, V.N.: An Overview of Statistical Learning Theory. IEEE Transactions on Neural Networks, Vol. 10. (1999) 988-999

Multi-attribute Decision Making in a Complex Multiagent Environment Using Reinforcement Learning with Selective Perception

Sébastien Paquet, Nicolas Bernier, and Brahim Chaib-draa

Laval University, Canada
{spaquet,bernier,chaib}@damas.ift.ulaval.ca

Abstract. Choosing between multiple alternative tasks is a hard problem for agents evolving in an uncertain real-time multiagent environment. An example of such environment is the RoboCupRescue simulation, where at each step an agent has to choose between a number of tasks. To do that, we have used a reinforcement learning technique where an agent learns the expected reward it should obtain if it chooses a particular task. Since all possible tasks can be described by a lot of attributes, we have used a selective perception technique to enable agents to narrow down the description of each task.

1 Introduction

Analyzing many possible tasks and choosing the best one is obviously a hard job for an agent, particularly in a dynamic and uncertain environment. In this case, an agent has to make a choice with only a partial view of the situation and it has to make it quickly, because the environment is constantly changing. To do that, the solution proposed here is to use a reinforcement learning algorithm in which the agent learns the expected reward it should get for every possible task it could be faced with.

In our test environment, the number of possible states is very large, since a task is described by many attributes which can take many possible values. To find the right level of precision for the state description, we have adapted the selective perception technique, developed by McCallum [1], to enable the agent to learn by itself which is the level of precision it needs in all possible situations.

Our test environment is the RoboCupRescue simulation in which the goal is to build a simulator of rescue teams acting in large urban disasters [2]. More precisely, this project takes the form of an annual competition in which participants are designing rescue agents trying to minimize damages, caused by a big earthquake, such as civilians buried, buildings on fire and blocked roads. In the simulation, participants have approximately 30 to 40 agents of six different types to manage: *FireBrigade*, *PoliceForce*, *AmbulanceTeam*, *FireStation*, *PoliceOffice* and *AmbulanceCenter*.

In this article, we focus only on *FireBrigade* and *FireStation* agents. The task of those agents is to extinguish fires. Furthermore, in order to be effective, they have to coordinate themselves on the same buildings on fire, because more than one agent is often needed to extinguish a building on fire.

A *FireBrigade* agent is faced with the problem of choosing a fire to extinguish between a list of buildings on fire. Since there could be a lot of fires, agents do not consider all fires at once. They separately choose which fire zone to extinguish and which specific building in the chosen fire zone to extinguish. Fire zones are simply groupings of near buildings on fire.

Since the *FireStation* agent can receive more messages, it normally has a better knowledge of the global situation compared to the *FireBrigade* agents. Therefore, it is the responsibility of the *FireStation* agent to allocate fire zones to *FireBrigade* agents. After they have received their fire zone, *FireBrigade* agents have the possibility to learn how to coordinate their efforts to extinguish the more important fires in a given fire zone.

To reduce the learning task complexity, each agent considers only one possible task at a time, i.e. agents are only considering one building at a time when choosing a building on fire. This enables us to have a fix state description. When an agent wants to choose a building on fire to extinguish, it goes through the list of all buildings on fire, it evaluates them independently by calculating the utility and the expected reward of each. The utility is an estimation of the importance to extinguish a given building, calculated by considering the number of civilians and buildings on danger. The expected reward, learned with the algorithm presented in section 2, can be seen as an estimate of the capacity to extinguish a given fire.

2 Selective Perception

To learn the expected reward of choosing one fire, we have used a selective perception technique [1], because the description of our states is too big. With this technique, the agent learns by itself to reduce the number of possible states. To do that, it uses a tree structure similar to a decision tree. At the beginning all states are considered to be the same, so there is only the root of the tree. After some experiences, the agent tests if it would be interesting to divide the different states, represented as the leaves of the tree. An advantage of this algorithm is that it can drastically reduce the number of states distinguished by the agent.

At each time step t, the agent records its experience captured as an "instance" that contains the action it made the step before (a_{t-1}), the observation it perceives (o_t) and the reward it obtains (r_t). Each instance also has a link to the preceding instance and the next one, thus making a chain of instances. In our case, we have one chain for each building that an agent chooses to extinguish. A chain contains all instances from the time an agent chooses to extinguish a building until it changes to another building.

Therefore, during the simulation, the agents record many instances organized in many instance chains and they keep all those instances until the end of the simulation. There is no learning taking place during the simulation, because the agents are evolving in a real-time environment and they cannot afford to take time to learn during the simulation. After the simulation, the agents are regrouping all their experiences together, the tree is updated with all those new instances and the resulting tree is returned to each agent.

To learn how to classify the states, we use a tree structure similar to a decision tree. The tree divides the instances in clusters depending on their expected reward. The objective here is to regroup all instances having similar expected rewards, because it means that the agent does not have to distinguish between them, since it should act the same way in all those situations.

The algorithm presented here is an instance-based algorithm in which a tree is used to store all instances which are kept in the leaves of the tree. The other nodes of the tree, called center nodes, are used to divide the instances with a test on a specific attribute. To find the leaf to which an instance belongs, we simply start at the root of the tree and head down the tree choosing at each center node the branch indicated by the result of the test on the instance attribute value. In addition, each leaf of the tree also contains a Q-value indicating the expected reward if a fire that belongs to this leaf is chosen. In our approach, a leaf of the tree l is considered to be a state for the reinforcement learning algorithm.

After a simulation, all agents put their new experiences together. This set of new experiences is then used to update the tree. Firstly, all the new experiences are added to their respective leaf nodes. Afterwards, the Q-values of each leaf node are updated to take into consideration the new instances which were just added. The updates are done with the following equation:

$$Q(l) \leftarrow R(l) + \gamma \sum_{l'} Pr(l'|l)Q(l') \tag{1}$$

where $R(l)$ is the estimated immediate reward if a fire that belongs to the leaf l is chosen, $Pr(l'|l)$ is the estimated probability that the next instance would be stored in leaf l' given that the current instance is stored in leaf l. Those values are calculated directly from the recorded instances:

$$R(l) = \frac{\sum\limits_{i_t \in I_l} r_{t+1}}{|I_l|}, \; Pr(l'|l) = \frac{|\{\forall i_t \in I_l | L(i_{t+1}) = l'\}|}{|I_l|} \tag{2}$$

where $L(i)$ is a function returning the leaf l of an instance i, I_l represents the set of all instances stored in leaf l, $|I_l|$ is the number of instances in leaf l and r_{t+1} is the reward obtained after the instance i_t was chosen.

After the Q-values have been updated, the algorithm checks all leaf nodes to see if it would be useful to expand a leaf and replace it with a new center node containing a new test, thus dividing the instances more finely. To find the best test to divide the instances, we try all possible tests, i.e. we try to divide the instances according to each attribute describing an observation. After all attributes have been tested, we choose the attribute that maximizes the error reduction as shown in equation 3 [3]. The error measure considered is the standard deviation ($sd(I_l)$) on the instances' expected rewards. If the standard deviation is reduced, it means that the rewards are closer to one another. Thus, the tree is moving toward its goal of dividing the instances in groups with similar expected rewards. The expected error reduction obtained when dividing the instances I_l of leaf l is calculated using the following equation where I_d denotes the subset of instances in I_l that have the d^{th} outcome for the potential test:

$$\Delta error = sd(I_l) - \sum_d \frac{|I_d|}{|I_l|} \times sd(I_d) \tag{3}$$

The standard deviation is calculated on the expected reward of each instances which is defined as:

$$Q_I(i_t) = r_t + \gamma Pr(L(i_{t+1})|L(i_t)) \times Q(L(i_{t+1})) \tag{4}$$

where $Pr(L(i_{t+1})|L(i_t))$ is calculated using equation 2 and $Q(L(i_{t+1}))$ is the value returned by equation 1.

As mentioned earlier, one test is tried for each possible instance's attribute. For a discrete attribute, we divide the instances according to their value and for a continuous attribute, we test different thresholds to find the best one.

Finally, after the tree has been updated, we update the Q-values again to take into consideration the new state space.

During the simulation, the agents are using the tree created offline to choose the best fire to extinguish. When choosing a fire to extinguish, the *FireBrigade* agent has a list of buildings on fire it knows about. For each building in the list, the agent finds the leaf of the tree to which this building belongs and record the expected reward associated with this leaf. The chosen building is the one that has the greatest expected reward.

3 Experimentations

As said before, experimentations have been done in the RoboCupRescue simulation environment. We have implemented the algorithm presented in this paper for the task of choosing a fire to extinguish by the *FireBrigade* agents. Each agent independently evaluates the list of buildings on fire it knows about and make its decision regarding which one to extinguish.

At first, the agent looks at the fires from a higher point of view. It only considers zones on fire, which are clusters of near buildings on fire. To make its decision, it should consider some attributes describing a zone on fire. In our experiments, there were almost 4×10^{13} possible states. After the *FireBrigade* agent has chosen a zone on fire, it should choose which fire to extinguish. To this end, this agent can examine the attributes describing a fire. In our experiments, there were more than 265×10^6 possible states. The results we have obtained show a drastic reduction of the state space. The agents have learned a tree with 998 leaves for the problem of choosing a zone on fire and 571 leaves for the problem of choosing a building on fire.

With those trees that are relatively small compared to the possible number of states, the agents were able to learn some interesting behaviors. In our preliminary tests, the agents were choosing useful zones and buildings on fire to extinguish. However, we also observed an unwanted variation of performances from one simulation to another. One hypothesis is that the variation is due to the uncertainty of the environment. There are a lot of hidden states because the agents are not able to perceive everything.

4 Related Work

The approach presented is inspired by the work of McCallum in his thesis [1]. In our work, we use approximately the same tree structure to which we have made some modifications. Firstly, we do not use the tree to calculate Q-values for every possible basic actions that an agent can take. We use the tree to calculate the expected reward if the agent chooses a certain task. It still has to find the actions to accomplish this task.

Another difference is in the way the leaves are expanded. Instead of generating all possible subtrees, we use an error reduction estimation measure, borrowed from the decision tree theory [3], that enables us to find good tests. Like Uther and Veloso [4] with their Continuous U-Tree algorithm, we also support continuous attributes, but with a different splitting criteria.

The way we manage our instance chains is also quite different. In our work, there is not only one chain, but many chains, one for each attempt to extinguish a fire or a fire zone. Our concept of instance chain is closer to the concept of *episode* described by Xuan, Lesser and Zilberstein [5].

5 Conclusion

This article has presented an efficient algorithm to enable the agent to learn the best task to choose, i.e. which building to extinguish. We have shown how we can adapt a reinforcement learning algorithm to the problems of task allocation and multi-attribute decision making in a complex application. Our algorithm enables the agents to manage a large state space by letting them learn by themselves to distinguish only the states that need to be distinguished. By doing so, the agent drastically reduces the number of states, thus facilitating the calculation of the Q-values. Our results show that agents are able to obtain good results with trees having a small number of leaves compared to the total number of states before learning. In our future work, we will adjust the agents behavior to take more into account the uncertainty in the environment and thus diminishing the variation in the results.

References

1. McCallum, A.K.: Reinforcement Learning with Selective Perception and Hidden State. PhD thesis, University of Rochester, Rochester, New-York (1996)
2. Kitano, H.: Robocup rescue: A grand challenge for multi-agent systems. In: Proceedings of ICMAS 2000, Boston, MA (2000)
3. Quinlan, J.R.: Combining instance-based and model-based learning. In: Proceedings of the Tenth International Conference on Machine Learning, Amherst, Massachusetts, Morgan Kaufmann (1993) 236–243

4. Uther, W.T.B., Veloso, M.M.: Tree based discretization for continuous state space reinforcement learning. In: Proceedings of the Fifteenth National Conference on Artificial Intelligence, Menlo Park, CA, AAAI-Press/MIT-Press (1998) 769–774
5. Xuan, P., Lesser, V., Zilberstein, S.: Modeling Cooperative Multiagent Problem Solving as Decentralized Decision Processes. Autonomous Agents and Multi-Agent Systems (2004) (under review).

Multi-agent Trail Making for Stigmergic Navigation

Alfred Wurr and John Anderson

Autonomous Agents Laboratory
Department of Computer Science
University of Manitoba
Winnipeg, Manitoba
Canada R3T2N2
{awurr, andersj}@cs.umanitoba.ca

Abstract. Robotic agents in dynamic environments must sometimes navigate using only their local perceptions. In complex environments, features such as terrain undulation, geometrically complex barriers, and similar obstacles form local maxima and minima that can trap and hinder agents using reactive navigation. Moreover, agents navigating in a purely reactive fashion forget their past discoveries quickly. Preserving this knowledge usually requires that each agent construct a detailed world model as it explores or be forced to rediscover desired goals each time. Explicit communication can also be required to share discoveries and coordinate actions. The cost of explicit communication can be substantial, however, making it desirable to avoid its use in many domains. Accordingly, in this paper we present a method of cooperative trail making that allows a team of agents using reactive navigation to assist one another in their explorations through implicit (*stigmergic*) communication.

1 Introduction

Robotic agents that employ reactive navigation suffer from several problems. Agents that are only reacting may be attracted to a local stimulus, but one that is really only a waypoint on route to a goal - a local maximum [1,2]. Box-canyons, for example, are a common local maxima problem, where agents are unable to see a goal of any type [3] and are reduced to wandering randomly before stumbling upon an exit [4]. On the other hand, when a goal is visible, these same agents may attempt to move in a direct line toward it, but be blocked by an interposing barrier - a local minimum [2]. Cyclic behaviour is another potential problem, where agents oscillate between multiple stimuli [5]. Compounding these problems, agents navigating in a purely reactive fashion, forget their past discoveries even as they are made (unless they are constructing a detailed world model as they explore) and as a result their explorations are unsystematic (they may cover the same ground repeatedly) [6].

Many researchers have attempted to remedy these and other problems using *stigmergy*, a term used in biology to describe the influence that previous changes to the environment can have on an individual's current actions [7]. To date, most work with stigmergy has dealt with its effects on simple homogeneous robots collectively performing foraging or sorting tasks (e.g. [8,7,1,9]) based on that of the natural creatures (e.g. ants and termites) that inspire the model. More sophisticated uses of stigmergy or stigmergy-inspired approaches have been proposed by Werger and Mataric [8,10], Balch and Arkin [4], Vaughan et al. [11] and Parunak, Sauter, et al., [9,12] to solve problems in navigation and other common tasks. Ant and other biologically-inspired forms of computation have also been studied in numerous other tasks (e.g. [13]).

2 Stigmergic Navigation

In this paper, we employ a number of different stigmergic techniques that involve the deployment and exploitation of physical markers (as opposed to markers used only internally and communicated explicitly) in order to assist in team navigation. We refer to the process of using these marking techniques collectively to make more purposeful reactive navigation decisions in an unknown environment as *stigmergic navigation*.

The primary method employed involves a team of agents constructing marker trails leading to discovered goal locations, cooperatively and dynamically. A number of simpler methods are used as additional supports (and which are effective on their own as well). These include marking bottlenecks, marking local maxima and marking local minima [14]. Marking bottlenecks involves creating marker trails through constricted areas (i.e. hallways and doorways), which draw agents to new areas. Local maxima markers assist agents in negotiating areas devoid of sensory stimulus by first attracting agents (drawing them from walls of rooms to areas of greater visibility) and then repelling them outward toward potential exits. Local minima markers are unique in that they prescribe an action to undertake in response to a specific situation at a particular place in the environment, rather than a more basic attractive/repulsive force. In our specific implementation, these markers are dropped when an agent perceives an attractor and is blocked by an interposing barrier. Agents react to local minima markers by jumping when sufficiently close to them. This has the effect of causing agents to avoid a specific type of local minima by leaping over the problem. Alternatively, reactions such as being repelled by local minima markers are also possible, but not explored in the experiments outlined here.

More detailed explanations of these supplemental marking techniques are available in [14]. The following section provides a detailed overview of stigmergic trail-making that is the focus of this paper.

2.1 Stigmergic Trail-Making

Stigmergic trail-making is a process whereby a team of agents implicitly *cooperates* in the construction of a purposeful trail of markers leading to a goal or goals while exploring their environment [14]. By following these trails, agents are then able to locate a previously discovered goal more rapidly on subsequent trips without benefit (or need) of internal world modelling, internal maps or internal path planning. Most notably, these trails are constructed, shared and used entirely without explicit communication.

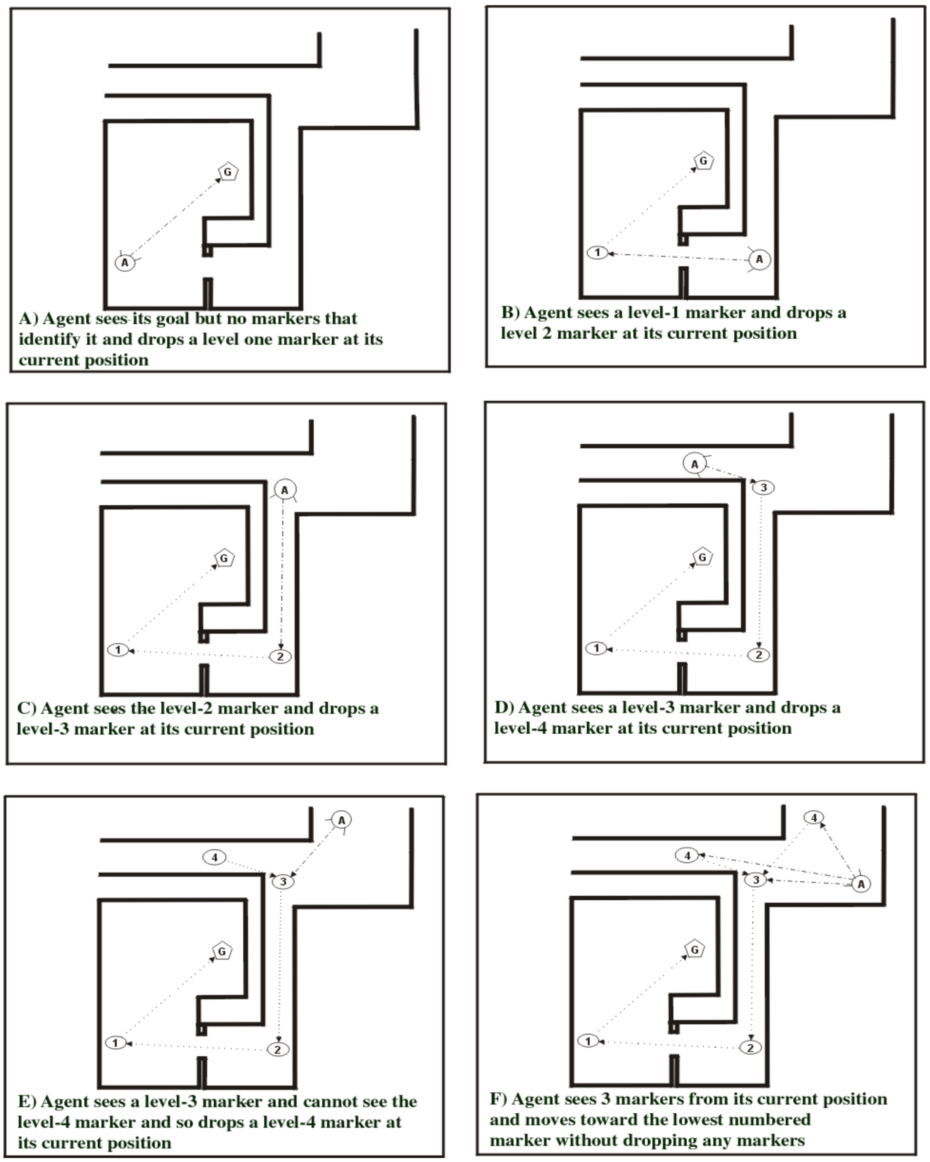

A) Agent sees its goal but no markers that identify it and drops a level one marker at its current position
B) Agent sees a level-1 marker and drops a level 2 marker at its current position
C) Agent sees the level-2 marker and drops a level-3 marker at its current position
D) Agent sees a level-3 marker and drops a level-4 marker at its current position
E) Agent sees a level-3 marker and cannot see the level-4 marker and so drops a level-4 marker at its current position
F) Agent sees 3 markers from its current position and moves toward the lowest numbered marker without dropping any markers
A
G
1
2
3
4

Fig. 1. Stigmergic trail making

The process of stigmergic trail-making is best illustrated by way of an example. Consider the situation in Figure 1. A primitive stigmergic trail is created when an agent perceives a goal and drops a marker on the ground at its current location (Figure 1a). The marker dropped is assigned a perceptible numeric value that represents its position in the trail (in this case, 1). The dropped marker identifies a vantage point from which the goal should be visible to an agent standing at the marker's location. By itself, a trail this rudimentary is of limited utility: to make the trail more sophisticated, it is necessary to extend the trail. As Figure 1 illustrates, this occurs when another agent (or the same agent at a later point in time) sees an end-point of the trail and drops a second marker (see Figure 1b). The second marker dropped is assigned a value one higher than the marker that the agent has observed (in this case, 2). As the process repeats, the trail gradually lengthens (see Figure 1c, d).

When the agent perceives the goal (or a marker already on the trail), it checks for the presence of stigmergic markers already serving as attractors to this target. If no markers are visible, the agent drops a marker as described above. This does not preclude the construction of multiple trails during this process. However, when such trails are made they will not be in obvious sight of the one another. This is illustrated in Figure 1e, where an agent sees the end of a trail, by perceiving a 3 marker but no 4, and so extends the trail in another direction. Note that under conditions such as this, extending the trail in different directions is very helpful - agents coming from different directions or from different sides of obstacles can find a useful path to a goal.

An agent only extends the trail when it perceives what appears to be its end. If the agent perceives other markers from its location, it only drops a marker if the goal or a higher valued marker is not also visible. If such a marker is visible, the agent knows it is not at the end of a trial and does not drop a marker. Such a situation is depicted in Figure 1f: the agent perceives marker 3 and two marker 4's; it knows that 3 is not the end of the trail (because it perceives a 4 marker), and that its vantage point is not far enough away to warrant dropping another marker (since existing markers already supply pertinent navigation information from the agent's current location, or the 4 marker would not be present).

These conditions allow an agent to simply follow a trail without dropping new markers when there is enough navigation information available to do so. Thus, a branching situation such as that shown in Figure 1e will only occur when an agent's perceptions indicate it is at the end of the trail when it in fact is not. In such a setting, there is not enough navigation information to conclude otherwise given the agent's current location, and so branching the trail at that point is a logical action.

Agents navigate to the goal at a trail's end by always moving toward the lowest numbered marker visible. Since the markers are dropped based on visibility, the agents are able to consistently follow one marker to the next to locate the goal. This also helps minimize the length of a path followed by an agent - if an agent perceives a 3 and a 4 marker, for example, the agent will not bother moving to the 4 marker.

The two important features to emphasize in this approach are a) that this trail-building occurs collaboratively; and b) that it occurs while agents follow existing trails and otherwise navigate through the environment.

In the case of several goals in close proximity, trails may connect. This may cause an agent to perceive more than one lowest valued marker: in this case an agent will be attracted to the marker in closest proximity, so may start following toward one goal and end up at another. Either way, a goal is reached.

3 Evaluation

Agents in this work were implemented using schema-based behaviours, which divide control into *perceptual* and *motor* schemas that are tailored to specific aspects of an agent's environment [15,2]. Perceptual schemas were created to detect the conditions under which markers should be dropped. In addition, motor schemas were implemented to cause agents to react appropriately to the presence of markers. Agent control was also composed of a number of other basic schemas (i.e. avoid-obstacles, wander) that were necessary to allow them to explore the chosen simulation environment; the computer game Half-Life a large-scale, 3D environment with realistic physics. The specific environment used was a standard Half-Life map, entitled Crossfire (a simplified 2-D overhead view can be found in [14]). To allow agents to mark their environment, a pre-existing game object (a small satchel) was adapted to function as a basic marker type that agents could drop on the ground as required.

Table 1. Summary of Results

Marking Method	Total Goals	Average Time	Average AgentInv
None	161	592.35	579822.88
BTl	278	463.67	43917.58
LocMax	224	533.48	52135.00
BTl/LocMax	305	567.02	51697.60
BTl/LocMax/LocMin	323	467.67	24068.98
StgTrl	1828	462.19	45043.30
StigTrl/BTl	2253	508.15	46896.00
StigTrl/BTl/LocMax	3817	525.3	30869.03
StigNav	3856	490.11	24814.38

None=No Markers, BTl=Bottleneck Markers, LocMax=Local Maxima Markers, LocMin=Local Minima Markers, StigTrl=Stigmergic Trails, StigNav=Stigmergic Navigation

Marker techniques were evaluated in a series of experiments in which the objective was for a team of six agents to find a specific goal at an unknown location in their environment as quickly and as many times as possible in a

30 minute period. Agents began each trial in one of two starting locations, so that agents would not interfere with one another excessively at the start of a trial. The goal itself was located in a room on the far side of the environment. After locating and moving to within 40 units (measured as a fraction of the total size of the environment, which is 8191 units cubed) of the goal, a point was added to the running total of goals discovered in the trial. This agent was then transported back to one of the two randomly selected starting rooms to attempt to find the same goal again. A set of 40 trials were run with a team of 6 agents that did not employ markers and combination of previously outlined marking methods. The environment, goal location and agent starting positions were consistent throughout all experimental trials for both teams.

Table 1 lists the total goals achieved across the 40 trials, the average real-world time (in seconds) and the average number of agent decision-making invocations (*AgentInv*) required to find the goal the first time in a trial. The number of agent decision-making invocations is intended as a secondary measure of time passing in the system. While the simple forms of stigmergy all improved navigation, stigmergic trail markers resulted in a very substantial improvement over agents using no markers, or even the improved performance gained using the first two marker types. The best performances, however, occurred when all four marker types were combined. Using all marker types in combination resulted in 3856 goals being reached in 40 trials for an average of 96.4 goals per trial (with a standard deviation of 60.62 goals). This is almost 23 times more goals than agents using no markers.

Stigmergic trail building was assisted by the other marker types because the bottleneck and local maxima markers encouraged agents to travel from area to area more frequently, causing agents to perceive and therefore extend the stigmergic trails more quickly. This allowed the team of agents to benefit from the trail for longer periods of time in each trial, leading to improved performance. The reason for such a large standard deviation in the number of goals in each trial is directly connected to the variation in how quickly the agents first locate the goal and start constructing the trail. In cases where stigmergic trails are built early in the trial, the team is essentially following this trail repeatedly until time expires. If the trail is not built early or at all, performance suffers correspondingly since agents have less time to benefit from it.

4 Summary

In this paper, we have described methods that reactively navigating agents can use to increase the frequency and ease with which they discover and subsequently navigate goals, without need of high-level path-planning, world modelling or explicit communication. While the use of stigmergy in general is not new, the development of specific techniques operating in a distributed fashion, such as that described here, encourages the application of stigmergy to real-world problems. In general, we believe the maintenance of knowledge in the world as opposed to inside an agent is strongly underappreciated in AI, and in future, techniques such as these will be applied more extensively and viewed as powerful tools.

References

1. Balch, T., Arkin, R.C.: Communication in reactive multiagent robotic systems. Autonomous Robots **1** (1994) 27–52
2. Pirjanian, P.: Behavior coordination mechanisms: State-of-the-art. Technical Report IRIS-99-375, Institute for Robotics and Intelligent Systems, University of Southern California, Los Angeles (1999)
3. Sgorbissa, A., Arkin, R.C.: Local navigation strategies for a team of robots. Technical report, Georgia Tech Robotics Laboratory (2001)
4. Balch, T., Arkin, R.C.: Avoiding the past: A simple but effective strategy for reactive navigation. In: Proceedings of the 1993 IEEE International Conference on Robotics and Automation. Volume 1. (1993) 678–685 Atlanta, Georgia.
5. Arkin, R.C., Balch, T.: Cooperative multiagent robotic systems. In Bonasso, R., Murphy, R., eds.: Artificial Intelligence and Mobile Robots. MIT/AAAI Press (1998) Cambridge, MA.
6. Moorman, K., Ram, A.: A case-based approach to reactive control for autonomous robots. In: AAAI Fall Symposium on AI for Real-World Autonomous Mobile Robots. (1992) Cambridge, MA.
7. Holland, O., Melhuish, C.: Stigmergy, self-organisation, and sorting in collective robotics. Artificial Life **5** (2000)
8. Werger, B.B., Mataric, M.: Exploiting embodiment in multi-robot teams. Technical Report IRIS-99-378, University of Southern California, Institute for Robotics and Intelligent Systems (1999)
9. Parunak, H.V.D., Brueckner, S., Sauter, J., Posdamer, J.: Mechanisms and military applications for synthetic pheromones. In: Workshop on Autonomy Oriented Computation. (2001) Montreal, Canada.
10. Werger, B.B., Mataric, M.: Robotic food chains: Externalization of state and program for minimal-agent foraging. In: Proceedings of the 4th International Conference on Simulation of Adaptive Behavior: From Animals to Animats 4, MIT Press (1996) 625–634
11. Vaughan, R., Stoy, K., Sukhatme, G., Mataric, M.: Lost: Localization-space trails for robot teams. IEEE Transactions on Robotics and Automation **18** (2002) 796–812
12. Sauter, J.A., Matthews, R., Parunak, H.V.D., Brueckner, S.: Evolving adaptive pheromone path planning mechanisms. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems, ACM Press (2002) 434–440 Bologna, Italy.
13. Brooks, R.: Cambrian Intelligence. MIT Press (1999)
14. Wurr, A.: Robotic team navigation in complex environments using stigmergic clues. Master's thesis, Department of Computer Science, University of Manitoba, Winnipeg (2003)
15. Balch, T., Arkin, R.C.: Behavior-based formation control for multi-robot teams. IEEE Trans. On Robotics and Automation **14** (1998)

A Decision-Theoretic Algorithm for Bundle Purchasing in Multiple Open Ascending-Price Auctions

Scott Buffett and Alastair Grant

Institute for Information Technology – e-Business
National Research Council Canada
46 Dineen Drive, Fredericton, New Brunswick, Canada E3B 9W4
{Scott.Buffett, Alastair.Grant}@nrc.gc.ca

Abstract. This paper presents an algorithm for decision-making where the buyer needs to procure one of possibly many bundles of complementary products, and items are sold individually in multiple open ascending-price (English) auctions. Auctions have fixed start and end times, and some may run concurrently. Expected utility of a bidding choice is computed by considering expected utility of choices at future decisions. The problem is modeled as a Markov decision process, and dynamic programming is used to determine the value of bidding/not bidding in each state. Three techniques for reducing the state space are given. Results show that a bidding agent using this algorithm achieves significantly higher utility on average when compared with one that does not consider the value of future choices.

1 Introduction

As the volume of e-commerce completed through online auctions rises, so does the need for efficient and effective decision-making systems that can analyze several options and help a potential buyer make rational bidding decisions. Online auction sites have grown considerably over the past few years by mainly targeting the consumer, but recent research has shown that more and more businesses are integrating online auctions into their supply chains [6]. While research in e-commerce focuses on the problems associated with finding and monitoring auctions that may be of interest to a potential buyer (or bidding agent), artificial intelligence techniques are needed to analyze the data and help the agent make effective decisions on which auctions to pursue as well as how much to bid.

In this paper, we present a decision-making algorithm for an agent that needs to purchase one of possibly many *bundles* of products, where items are sold individually in auctions. In our context, we consider a bundle to be a user-defined set of complementary products. There also may be alternatives for certain products, and consequently several acceptable bundles. We consider open ascending-price (English) auctions where the start time, finish time and opening bid are fixed and known in advance. There is no restriction on auction times (i.e. time periods for some auctions may overlap). The goal is to analyze the incomplete information on current and future auctions and make bidding decisions that give the agent the best chance of ultimately procuring the best bundle in terms of bundle preference and cost.

Previous work [2,5] has analyzed the problem of determining the expected utility over sets of auctions, but this work bases the decision on whether or not to participate in an auction on whether or not the auction is part of the set deemed the best in terms of expected utility at that time. This “greedy” approach works best when a buyer immediately must commit to a particular set. In our setting, we compute the expected utility of a choice by considering the expected utility of future consequential decision points. This is the main contribution of the paper. To accomplish this, we use a purchase procedure tree [1] to model the system of decisions and purchases, and model the decision problem as a Markov decision process. Dynamic programming is then used to determine the optimal choice at each decision. To reduce the consequentially unmanageable size of the resulting state space, we employ three state-reducing techniques. Results show that our method performs significantly better than the greedy approach in a specific example.

2 Problem Formalization

Let $\mathcal{A}$ be a set of English auctions where each $a \in \mathcal{A}$ is an auction for product p_a, and all auctions in $\mathcal{A}$ are currently open or will open in the future (i.e. none are finished). Let $P = \{p_a \mid a \in \mathcal{A}\}$ be the set of products to be auctioned in $\mathcal{A}$, and let $\mathcal{B} \subseteq 2^P$ be a set of bundles. Each bundle is specified by the buyer as being a satisfactory and complete set of products. To specify the buyer's preferences assume that a bundle purchase utility function $u : \mathcal{B} \times C \to \Re$ is given, where C denotes the set of possible bundle costs. The problem is to decide whether or not to bid in each auction, with the goal of ultimately obtaining all products in some bundle $b \in \mathcal{B}$ at cost c with the highest utility possible.

3 The Purchase Procedure Tree

In order to structure the decision process, we use a purchase procedure tree (PPT) [1]. This tree graphically depicts the process of decisions and auctions that, when executed, will result in a complete bundle purchase. Starting at the root node, the buyer proceeds toward the leaf nodes, bidding in auctions at auction nodes and making decisions at decision nodes. At each decision node (lower case d's) there are two choices: participating in the auction that will end next (always represented by the left child of the decision node), or allowing it to pass. When the auction ends, if the buyer is the winner then execution moves to the left, else to the right. Once a path is completed, the buyer will have procured a complete bundle. The example PPT in Figure 1 represents the problem where there are bundles AB, AC, BD and EF, and the auction for A ends first. The current decision (d_1) to be made is on whether or not to bid on A. The PPT shows the consequential decisions and auctions which result from each choice. Note that a new PPT is built for each decision. For example, if A is purchased then a new PPT will be built with root d_3. This tree may include new options, and perhaps may even include bundles that do not include the purchased items (i.e. BD or EF) if they are deemed viable options.

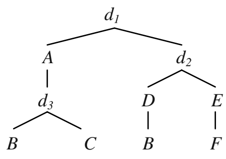


Fig. 1. An example purchase procedure tree

4 The MDP Model

To determine the expected utilities of each option, the sequence of auctions and decisions is modeled as a Markov decision process (MDP), and the optimal policy in the MDP is determined using the value iteration method of dynamic programming. Each state in the MDP is a 5-tuple $\langle P, c, \mathcal{A}_{cur}, C_{\mathcal{A}_{cur}}, t\rangle$ where P is the set of purchased products, c is the total amount spent on purchases, $\mathcal{A}_{cur} = (a_1, \ldots, a_m)$ contains the auctions that are currently running, $C_{\mathcal{A}_{cur}} = (c_1, \ldots, c_m)$ contains the current bid c_i for each a_i, and t is the time. The set of actions is $Q = \{bid, notbid\}$. Each terminal state has an associated reward, equal to the utility $u(b, c)$ of purchasing the bundle $b = P$ at cost c. The value $v(s)$ for a state s is computed as the expected utility of s. The problem is to determine $v(s)$ for each reachable state s in order to find the optimal policy $\pi : S \rightarrow Q$. For a state s' at which a bidding decision must be made, $\pi(s')$ advises which course of action maximizes expected utility. Because of the stochastic nature of auctions, for many actions it is not known for certain what the next state will be. However, we assume that the buyer will have some idea of what the outcomes of an auction will be (by examining auction history, market history, estimation of competitors reserve values, etc.). We model this information in the form of a *prediction function* $F_a(c, t, t')$. For an auction a, $F_a(c, t, t')$ takes a bid c and times t and t' (where $t < t'$), and returns a probability distribution p on the outcomes for the current bid at time t' given that the current bid at time t is c.

The problem with modeling the decision problem in this way is that there will be far too many states in the MDP. At any given time, there may be several auctions open, each of which with several possible outcomes for the current bid. Also, there may be several different combinations of items already purchased by the buyer, and several possible costs for those purchased goods. An important contribution of this paper lies in how we deal with this computational complexity without losing too much of the important information. In particular we do three things to reduce the state space:

1. *Assume that bidding is only done at the end of the auction.* With this assumption, only states that occur when an auction ends are considered in the MDP. At these points the agent chooses either to bid (thus winning the auction), or pass. Realistically, with this strategy the bidder runs the risk of not having a bid accepted before the deadline if there are one or more bidders with the same strategy. But this is not necessarily the true strategy to be used by the buyer. It is only the assumption made about future actions in the MDP model to ease the computational burden. The buyer is free to bid earlier in an auction if the expected utility of doing so is higher than the expected utility of waiting until the end. As a result, since the utility of winning an auction with an earlier bid is always at least as good as winning it with a later bid (since bids never go down), the buyer's true expected utility is at least as high as that predicted by the algorithm (given that the prediction functions are sufficiently accurate).
2. *Use the purchase procedure tree.* The PPT shows the sequence of decisions and auctions that follow from any choice. We use this to limit the state space in two ways. First, two auctions a_1 and a_2 are considered together in A_{cur} in a state only if there is a common ancestor decision node d (of the nodes representing a_1 and a_2) in the PPT such that a_1 and a_2 will be open when the decision at d must be made. Second, for any state s at node n in the PPT, the set P in s is always equal the union of the set of ancestor purchases in the PPT and the set of purchases already made before the PPT was built.
3. *Use the Pearson-Tukey three-point discrete approximation.* The probability measure p given by $F_a(c, t, t')$ assigns positive probability to only three values, according to the Pearson-Tukey three-point approximation [3,4]. Specifically, $p(x_1) = .185$, $p(x_2) = .63$ and $p(x_3) = .185$ where $Prob(X > x_1) = .95$, $Prob(X > x_2) = .5$ and $Prob(X > x_3) = .05$.

The transition probability function $Pr(s'|s, q)$ takes states s and s' and an action q and returns the probability of occupying s' directly after q is performed in s. Dynamic programming is then used to find the value of each state:

$$v(s) = \begin{cases} u(P, c) & \text{if } P \in \mathcal{B} \\ \max\{\sum_{s' \in S} v(s')P(s'|s, bid),\ \sum_{s' \in S} v(s')P(s'|s, notbid)\} & \text{otherwise} \end{cases} \quad (1)$$

and $\pi(s) = bid$ if $\sum_{s' \in S} v(s')P(s'|s, bid) > \sum_{s' \in S} v(s')P(s'|s, notbid)$, and $\pi(s) = notbid$ otherwise.

5 Results

The utility achieved using our method was compared with the utility achieved using the greedy method that instructs the buyer to bid on an item if it is part of the bundle with the highest expected utility. Tests were run using the product set $P = \{A, B, C, D, E, F\}$ with auction times $A : [0, 10]$, $B : [15, 45]$, $C : [15, 35]$, $D : [20, 40]$, $E : [25, 55]$ and $F : [38, 60]$, and the bundle set $\mathcal{B} = \{AB, CD, EF\}$. Each agent had the same preferences for bundles and were risk-neutral. 2000 tests were run for each bidding method, of which 1298 instances saw the two agents purchase a different bundle. Only these cases are examined. In each test run, the bidding agent being tested competed against 8 dummy bidders in each auction. Dummy agents' reserve values were chosen at random from reasonable distributions. Before the tests were run, the bidding agents were allowed to observe the dummy agents in order to learn the prediction function for each auction. Results showed that the mean utility achieved by our agent was 0.49079 (95% confidence interval $[0.48902, 0.49257]$), compared with 0.48607 ($[0.48468, 0.48747]$) by the greedy agent. The mean of the paired differences was 0.00472 ($[0.00242, 0.00702]$). A paired t-test indicates that the difference in the means is significant at $p < 0.0001$. While this is certainly not an exhaustive test, it shows promise that, in at least one scenario, our method significantly outperforms simple greedy decision-making.

6 Conclusions and Future Work

This paper presents an effective algorithm for making bidding decisions in multiple English auctions where the buyer needs to procure one of possibly several bundles of products. Expected utilities of choices are estimated more accurately by considering the value of future decisions. This results in better decision-making and higher utility. For future work, we plan to carry out more extensive experiments to further support our claims. One idea is to test our method on sets of actual online auctions, such as those found on *eBay*. This will involve monitoring auctions for a period of time in order to determine a prediction function for similar future auctions, and then simulating these real auctions with our bidding agent. This will give an idea not only of how well our technique performs in real auctions, but also how accurately these prediction functions can be determined.

References

1. S. Buffett and B. Spencer. Efficient monte carlo decision tree solution in dynamic purchasing environments. In *Proc. International Conference on Electronic Commerce (ICEC2003)*, pages 31–39, Pittsburgh, PA, USA, 2003.
2. A. Byde, C. Preist, and N. R. Jennings. Decision procedures for multiple auctions. In *Proc. 1st Int Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'02)*, pages 613–620, Bologna, Italy, 2002.
3. D. L. Keefer and S. E. Bodily. Three point approximations for continuous random variables. *Management Science*, 29(5):595–609, 1983.
4. E. S. Pearson and J. W. Tukey. Approximating means and standard deviations based on distances between percentage points of frequency curves. *Biometrika*, 52(3-4):533–546, 1965.
5. C. Preist, C. Bartolini, and A. Byde. Agent-based service composition through simultaneous negotiation in forward and reverse auctions. In *Proceedings of the 4th ACM Conference on Electronic Commerce*, pages 55–63, San Diego, California, USA, 2003.
6. T. Smith. eBay and your IT supply chain. TechWeb's Internet Week, April 12, 2002, url = "http://www.internetweek.com/story/showArticle.jhtml?articleID=6406064" (accessed on December 10, 2003), 2002.

The Use of Increasingly Specific User Models in the Design of Mixed-Initiative Systems

Michael Fleming

Faculty of Computer Science, University of New Brunswick
540 Windsor Street
Fredericton, NB E3B 5A3
mwf@unb.ca

1 Introduction

Work in the field of mixed-initiative interaction [1,2,3] suggests a more flexible approach to the reasoning process in artificial intelligence systems. In a mixed-initiative system, both the computer and the user can play an active role in a problem-solving session. At any given time, either party might take control of a session. The primary goal behind a mixed-initiative system is to take advantage of the fact that computers and people have very different strengths when it comes to solving problems.

One ideal feature of a mixed-initiative system is the ability to make intelligent decisions about *whether or not* it is appropriate to seek further assistance from a user at any given time. In [4], we propose that a system should solicit further input from a user precisely when the perceived benefits of this interaction exceed the expected costs. We then present a specific approach for modeling these benefits and costs. With this model clearly defined, it will be possible to design mixed-initiative systems, for any application area, that clearly stipulate when the system should take the initiative to interact with the user.

2 Modeling User Knowledge

A user model is a system's internal representation of a user's knowledge, abilities, preferences, goals and any other user-specific information that helps a system to adapt its behaviour in a way that is appropriate for each user. The user model is a crucial component of our model for reasoning about interaction with users.

In this paper, we focus specifically on one of the essential elements in our design: the ability to model the probability that a given user will have the knowledge required to answer a particular question. We refer to this probability value as P_{UK}. The idea is for the system to make different decisions about interaction in different situations, based in part on the identity of the specific user involved in the current session and on the system's perception of that user's expertise.

Our model relies on the use of increasingly specific user models as the system gains familiarity with each individual user. For example, when a user is unknown to the system, a very general user model is employed, one that contains information about the knowledge expected of all users or of users belonging to a particular *stereotype*.[1] However, as the system becomes more and more familiar with a user, it attaches more and more weight to the specific user model that it has built for that user.

As a system interacts with different users, it will often have to update its user models to reflect the new evidence that it has acquired. For example, a system that initially believed a user to be knowledgeable about a particular topic, but discovered that the user was twice unable to answer questions about that topic, should update its beliefs about that user to reflect this new information.

In determining the likelihood of a user knowing the answer to a particular question, a system might want to incorporate different types of information about user knowledge. This includes information about the *specific* user's knowledge, about the knowledge expected of users in the same stereotype, and about the general knowledge expected of *all* users. For each of those classes of information, a system might have expectations about the probability of getting an answer to the *specific* question being considered, about a particular *class* of questions, or about *any* question at all in the domain.

In this section, we provide a matrix-based framework that can be used to determine, at any moment in time, a single P_{UK} value representing the probability of the user being able to answer a particular question. This is done by considering all of the types of information described in the previous paragraph, weighted according to the amount of experience the system has with particular users and topics and according to initial weights specified by the system designer. We will describe how information about user knowledge is stored and how it is used to compute P_{UK}.

Suppose that a system is aware of m distinct users and n stereotypes to which users might belong. Suppose also that we are able to enumerate i possible questions or topics that a system might want to ask of a user, and j classes into which these questions can be categorized.

Our framework then requires the following structures:

– For each user, construct a $1 \times (m + n + 1)$ vector $\mathbf{u}$. The first m entries are used to identify the user as one of the m users about which the system has a profile. These m entries include a single 1 and $m - 1$ zeroes. In the example vector below, the user is the second of four known users. The next n entries classify the user into one or more stereotypes. If the system designer wishes for each user to belong to only one stereotype, then these n entries will include a single 1 as the only non-zero number. However, it is also possible to indicate that a user possesses some characteristics of more than one stereotype by including multiple non-zero entries that sum to 1. In the example below, there are three stereotypes; the user is primarily identified

[1] "A stereotype represents a collection of attributes that often co-occur in people" [5]. The idea is to use this type of information to allow systems to predict the preferences or knowledge of a user based on general characteristics of a class of users to which this user is believed to belong.

as belonging to the first stereotype, but has some properties of the third as well. The final entry of this vector is always 1 and indicates that the user is a member of the class of "all users" of the system.

$$\big(0\ 1\ 0\ 0 \mid 0.9\ 0\ 0.1 \mid 1\big)$$

Also for each user, construct a second $1 \times (m+n+1)$ vector $\mathbf{uw}$. This vector stores weights to indicate how much impact each *type* of user-knowledge information should have in computing P_{UK}. Initially, the system might have no information at all about individual users, in which case the weight on the user-specific information should be zero.[2] In these early situations, predictions about the knowledge of a user will be based entirely (or mostly) on stereotypical information and estimates of the knowledge expected of *any* user. The actual weights for stereotypes and for the "all-users" entry should be set up according to the desired contribution of each. For instance, if the system designer is confident that a user's stereotype will be a very accurate predictor of the user's behaviour, then she might set the initial weight on the stereotypical information to be quite high relative to the weight of the "all-users" entry (say, 0.9 vs. 0.1), as shown in the example vector below. If, on the other hand, there are no stereotypes defined or there is low confidence in their relevance to any given user, the information about *all users* should carry most of the weight.

$$\big(0\ 0\ 0\ 0 \mid 0.9\ 0.9\ 0.9 \mid 0.1\big)$$

The weights in this vector will be updated as the system makes its own observations about the knowledge of users, again according to the specifications of the system designer. For example, once the system has observed the user's expertise on a particular topic on a few occasions, this user-specific information should be the primary predictor of whether the user will be helpful on this topic in the future. In other words, the weight of the contribution from the user-specific information should increase gradually until it eventually overwhelms the weight on the stereotype and "all-users" contributions.

– For each possible topic or question, construct two $(i+j+1) \times 1$ vectors $\mathbf{t}$ and $\mathbf{tw}$. These are analogous to the user vectors described above. In the first vector, the first i entries will uniquely identify the question, the next j entries will classify it into one or more classes of questions, and the final entry will always be one. The example below shows the topic vector for the third of five questions, classified as belonging to the second of two question types.

$$\big(0\ 0\ 1\ 0\ 0 \mid 0\ 1 \mid 1\big)^T$$

[2] In some systems, initial interviews might be used to gather some concrete information about individual users, in which case some weight would be placed on the user-specific information that is gathered.

The second vector **tw** will indicate the desired weights on specific questions, general topic areas, and the domain as a whole.

– Construct a large $(m + n + 1) \times (i + j + 1)$ matrix **PK**. This matrix stores probability values: each entry represents the likelihood that a specific user, a class of users, or any user at all would know the answer to a specific question, a question of a certain type, or any question in the domain. The rows identify the user or user stereotype, while the columns represent questions or topic areas. In the matrix below, for example, the bold entries indicate that the first user has a probability of 0.8 of being able to answer the first question, and that users belonging to the third stereotype have a probability of 0.7 of answering questions of the second type.

$$\left(\begin{array}{ccccc|cc|c}
\mathbf{0.8} & 0.6 & 0.5 & 0.2 & 0.0 & 0.7 & 0.3 & 0.7 \\
0.7 & 0.6 & 0.7 & 0.2 & 0.0 & 0.7 & 0.3 & 0.6 \\
0.7 & 0.6 & 0.7 & 0.2 & 0.0 & 0.6 & 0.3 & 0.5 \\
0.7 & 0.6 & 0.7 & 0.2 & 0.0 & 0.6 & 0.3 & 0.6 \\
\hline
0.7 & 0.6 & 0.7 & 0.2 & 0.0 & 0.8 & 0.5 & 0.6 \\
0.8 & 0.7 & 0.8 & 0.2 & 0.0 & 0.6 & 0.6 & 0.7 \\
0.9 & 0.8 & 0.9 & 0.2 & 0.0 & 0.4 & \mathbf{0.7} & 0.6 \\
\hline
0.8 & 0.7 & 0.8 & 0.2 & 0.0 & 0.6 & 0.6 & 0.6
\end{array}\right)$$

Now, if the system is considering whether or not to ask a user a question, it can use matrix multiplication to determine a single P_{UK} value, based on a weighted combination of the above information.

This is done by multiplying **u** by **uw** element-wise, to yield a new vector $\mathbf{u_{wtd}}$, multiplying **t** by **tw** element-wise, to yield a new vector $\mathbf{t_{wtd}}$, and then using matrix multiplication to obtain $P_{UK} = \mathbf{u_{wtd}}\ \mathbf{PK}\ \mathbf{t_{wtd}}$.

In practice, when the vectors and matrices are very large, it will make sense to perform these multiplications not with the entire matrices, but by selecting only the rows and columns that are relevant to the current user and the current question. Examples of the use of this method can be found in [4].

The actual probabilities of different users being able to answer different questions and the weights on the various components of the model are also maintained through the use of a set of matrices and vectors. A matrix $\mathbf{N_{ask}}$ tracks how many times each question has been asked of a user. Another matrix, $\mathbf{N_{ans}}$, tracks how many times the user has actually provided an answer to each question.[3] The values in these matrices are used to compute the individual values stored in the matrix **PK**. Note that as weight vectors change over time, placing more weight on specific questions, the P_{UK} value itself will increase if the user has been found to be able to answer these questions. Details on updating probability values and weights can be found in [4].

[3] The model also allows the system to keep track of the *correctness* of a user's responses, in domains where the correct answer can be determined after the fact.

3 Discussion

It is very important for a mixed-initiative system to maintain a model of the likelihood that users will possess certain types of knowledge. We have provided a concrete quantitative approach to merging information that the system has available about the knowledge of specific users, of different classes of users and of all users in general, as well as about different classes of questions that might be asked. This type of a hybrid method is a novel approach to modeling knowledge about users and will be of interest to the user modeling community.

Although the idea of using user-specific information in combination with stereotypical information is certainly not new (see [6], for example), the traditional approach has been to have any user-specific information that has been gathered override the default, stereotypical assumptions of the system.

However, in many applications, it is not possible to determine that a user *does* or *does not* know about a particular topic. We cannot conclude from the fact that a user correctly answered one question in a given topic area that the user is knowledgeable on that subject. Using the model presented in this paper, a system can *gradually* incorporate the specific user models that it has constructed for each user. Furthermore, the system designer can control how quickly the system will change from relying strictly on its default, stereotypical assumptions to relying on the user-specific information it has gathered.

The work of Vassileva et al. [7] is also relevant because it describes the process of modeling a user by integrating information from different sources. Our framework could also be used for such a purpose, by weighting the contributions of different agents just as we currently weight the contributions of user-specific models and more general models.

Acknowledgement. The author wishes to acknowledge the helpful suggestions of Robin Cohen during the preparation of this paper.

References

1. Haller, S., McRoy, S., eds.: Papers from the 1997 AAAI Symposium on Computational Models for Mixed Initiative Interaction. AAAI Press (1997)
2. Cox, M., ed.: Proceedings of the 1999 AAAI Workshop on Mixed-Initiative Intelligence. Menlo Park, CA: AAAI Press (1999)
3. Haller, S., Kobsa, A., McRoy, S., eds.: Computational Models of Mixed-Initiative Interaction. Dordrecht, Netherlands: Kluwer Academic Publishers (1999)
4. Fleming, M.: Reasoning about interaction in mixed-initiative artificial intelligence systems. Technical report, University of Waterloo Ph. D. thesis (2003)
5. Rich, E.: Stereotypes and user modelling. In Kobsa, A., Wahlster, W., eds.: User Models in Dialog Systems. Springer-Verlag (1989)
6. Chin, D.N.: KNOME: Modeling what the user knows in UC. In Kobsa, A., Wahlster, W., eds.: User Models in Dialog Systems. Springer-Verlag (1989) 74–107
7. Vassileva, J., McCalla, G., Greer, J.: Multi-agent multi-user modeling in I-Help. User Modeling and User-Adapted Interaction **13** (2003) 179–210

Evaluating a Smart Recommender for an Evolving E-learning System: A Simulation-Based Study

Tiffany Tang[1, 2] and Gordon McCalla[2]

[1] Department of Computing, Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong
cstiffany@comp.polyu.edu.hk

[2] Department of Computer Science, University of Saskatchewan
Saskatoon, Saskatchewan, Canada
mccalla@cs.usask.ca

Abstract. In this paper we discuss paper recommendation techniques for learners in an evolvable e-learning system. We carried out an experiment using artificial learners for two techniques. The first one is based on the matching of learner model to the papers (pure model-based recommendation). And the second technique is based on peer learner recommendation (hybrid collaborative filtering), which is relatively domain independent. Experimental results show that hybrid collaborative filtering, which we believe can lower computational costs, will not compromise the overall performance of the recommender system.

1 Introduction

In e-learning, making recommendations of learning objects in a pedagogically ordered list is very important, quite different from single product recommendations in e-commerce. For example, some instructors will recommend students to read an interesting magazine article before a technical paper, because it will help students understand the technical paper or make them less intimidated. In this paper we will discuss our work on recommending research papers for learners engaged in an evolving web-based learning system [3]. In particular, we will focus on how to recommend and present pedagogically suitable reading materials based on a careful observation of learner interactions with the system; and also show through an artificial-learner-based simulation the effect and benefits of introducing pedagogical similarity in this domain.

There are several related works in recommending technical papers. Basu et al. [1] studied recommendation in the context of assigning conference papers to reviewing committee members. Woodruff et al. [4] discuss an enhanced digital book with a spreading-activation-geared mechanism to make customized recommendations for readers with different type of background and knowledge. McNee et al. [2] investigate the adoption of collaborative filtering techniques to recommend papers for researchers. These works are different from ours in that we not only recommend papers according to learners' interests, but also pick up those *not-so-interesting-yet-pedagogically-suitable* papers for them. Pedagogically valuable papers might not be interesting and papers with significant influence in the research community might not be pedagogically suitable for learners.

2 Our Approach: Pedagogy-Oriented Paper Recommendation

Our goal can be stated as follows:

> *Given a collection of reading materials and a learner's characteristics, recommend and deliver a set of materials in a pedagogically appropriate sequence, so as to meet both the learner's pedagogical needs and interests.*

The system must select some highly authoritative and (individually) interesting materials. The curriculum also requires a broad coverage of reading materials, including those not-interesting-yet-pedagogically useful. In this paper, we only consider a simplicity-based measurement among reading materials. The similarity criterion is used for two purposes: avoiding redundancy and recommending an item. Moreover, in our current simulation we only consider t-ordering (technical level) and d-ordering (pedagogical value) in delivering the reading materials[1]. We assume that all artificial learners do not have preference towards the length of the material, the abstraction of presentation, the prestige and the year of publication. We will explore more about other orderings in the future. Ideally, the recommender system will maximize a learner's utility such that the learner gains a maximum amount of knowledge and is well motivated at the end of the course. Intuitively, a model-based recommendation through a careful assessment of learner characteristics is very costly; because when new reading material is introduced, detailed identification is required, and when a learner gains some new knowledge after reading a paper, a re-matching is required to find the next suitable reading material for him/her. Alternatively, we can use a collaborative filtering technique (CF) to reduce the complexity of the recommendation process. Therefore, the matching process is not from learner model to learning material, but from learner model to other learner models. Since the system also utilizes some characteristics of learning material, it is a hybrid-CF. The remaining question is whether or not the hybrid-CF is as good as the model-based one.

3 Evaluating the Hybrid Recommendation Technique

3.1 Simulation Setup

We first generate 500 artificial learners and use 50 papers related to data mining for the learning material. The recommender system then recommends 15 papers to each learner according to each individual learner model (pure model-based). Each learner rates these papers according to their properties. After that, we generate 100 additional artificial learners, who are the target learners. Then, two recommendation techniques are applied to these target learners so as to evaluate their differences and performances. The first technique is the same as the technique used in the first 500

[1] For the definition of similarity and ordering, please refer to [3].

learners, i.e. model-based recommendation. The second technique is a hybrid-CF. In the simulation, we use the following minimal learner properties to generate artificial learners: learner ID #, background knowledge [k_1, k_2, …, k_{10}], learner interest toward specific topics [I_1, I_2, …., I_{12}], and learner motivation $M \in [0, 1]$. Where M = 1 represents that the learner is willing to spend more time to learn something not covered/understood before, and M = 0 represents that the learner is unwilling to explore any piece of knowledge not covered in the paper. Moreover, we use the following properties for the papers: paper ID #, technical knowledge [k_1, k_2, …, k_{10}], topics [I_1, I_2, …., I_{10}], and paper authority level. k_i denotes the depth of the knowledge of topic I_i used inside the paper. Deep knowledge means that a learner needs a good background in the corresponding knowledge in order to be able to understand the paper thoroughly. If the learner lacks that knowledge, then he/she/it can gain the corresponding knowledge by reading the paper carefully and spending more time to look at the references.

In addition, five core papers, i.e. papers that are pedagogically required for all learners, will be recommended regardless of learners' interests or knowledge; at least two papers with high technical level should be recommended to the learner; and in total 15 papers must be read and rated by each learner. The above requirements define the constraints on recommendation, and differentiate recommendation in e-learning from other application areas.

3.2 Model-Based Recommendations

Model-based recommendation is based on the following rules (*learner-centric*): The system starts with recommending acceptable papers in terms of a learner's knowledge level and the similarity of learners' interest towards the topic in the paper. Then, two interesting papers, but with very high technical level will be recommended, in order to improve the learner's knowledge. Finally, some not-interested-yet-pedagogical-useful (authoritative) papers will be provided as the part of the learning requirement in the end. After learners finish a paper, some additional knowledge may be acquired, depending on learner motivation. In our simulation, we assume that the increment of the learner's knowledge is based on the following rule.

IF $paper.k_j > learner.k_j$ **AND** $paper.authority$ = TRUE **THEN**

$learner.k_j = (paper.k_j - learner.k_j) \times learner.M \times Interest \times w_1 + learner.k_j$

IF $paper.k_j > learner.k_j$ **AND** $paper.authority$ = FALSE **THEN**

$learner.k_j = (paper.k_j - learner.k_j) \times learner.M \times Interest \times w_2 + learner.k_j$

where w_1 and w_2 represent factors that might affect learning speed after reading an authoritative/non-authoritative paper. These are two control variables in our experiment. *Interest* represents the similarity of learner interest to the paper's topic which due to limited space will not be described here. Moreover, the *total gain* made by the learner is defined as the value added from reading the paper. After a learner reads a paper, we need rating-generation rules to generate learner rating toward the paper, which will not be described here. If the rating is high, learner motivation will increase randomly with increasing rate $\boldsymbol{x}$. If the rating is low, learner motivation will decrease randomly with decreasing rate $\boldsymbol{x}$. $\boldsymbol{x}$ is another control variable that represents how much motivation a learner gains/loses after reading a paper.

3.3 Hybrid Recommendations

For each target learner, we find five neighbors based on the similarity of their interest and background knowledge. From these five nearest neighbours, we can get a set of candidate papers based on their ratings. Then, we order these papers from the highest ratings for the closest neighbour to the lowest rating for the furthest neighbor. Then the system will recommend up to eight authoritative papers starting from those receiving the highest rating followed by recommending non-authoritative papers. Then, the system will choose and recommend two very interesting and highly technical papers, and recommend five pedagogically required papers, if the learner has not read them. Finally, the system recommends the rest of the papers according to the rating order, until up to 15 papers in total.

3.4 Evaluation Metrics and Control Variables

There are two metrics recorded in the simulation, i.e. average learner motivation after recommendation and average learner knowledge after recommendation. And for the purpose of comparison, we compare the percentage differences between model-based recommendation and hybrid recommendation. In our simulation, control variables w_1, w_2 and $\boldsymbol{x}$ are adjusted to differentiate artificial learners as follows. $\boldsymbol{x} = 1$ for fast motivation change (FMC), $\boldsymbol{x} = 0.3$ for moderate (MMC), $\boldsymbol{x} = 0.1$ for slow (SMC), and $\boldsymbol{x} = 0$ (NMC). Moreover, we use eight pairs of (w_1, w_2), which are (1, 0), (U[0, 1], 0), (U[0, 0.3], 0), (1, U[0, 0.3]), (U[0, 1], U[0, 0.3]), (U[0, 0.3], U[0, 0.3]), (1, U[0, 1]), (U[0, 1], U[0, 1]), (1, 1), where U[0, y] means a random value generated from a uniform distribution function. The pair value represents the effect of authoritative and non-authoritative papers in the increment of the learner's knowledge. For example, (1, 0) indicates that only authoritative papers can fully increase a learner's knowledge. And (1, 1) indicates that both authoritative and non-authoritative papers are equally weighted and can fully increase a learner's knowledge. Each group of experiments is repeated thirty times for statistical analysis.

4 Experimental Results and Discussion

Table 1 shows the result of the experimentation. The value shown in each cell is the pair value of the percentage difference between model-based recommendation and the hybrid-CF technique in terms of average learner knowledge and motivation. A negative value indicates that the model-based recommendation technique is better than hybrid-CF. And a positive value represents the reverse situation. For example, the pair value (**0.65; 2.93**) represents that using hybrid-CF is 0.65% and 2.93% better than using model-based in terms of the average learner knowledge and motivation respectively. All results are checked by t-test for equal mean hypothesis (assuming different variance). The value in italics inside the table shows that the null hypothesis is not rejected (for $\alpha = 0.05$), or the difference between model-based and hybrid-CF is not statistically significant. If we exclude zero and italic values in Table 1, then there are 16 and 8 negative values for the difference of learner knowledge and motivation respectively, with the lowest value equals to -1.36% and -4.53% respectively. And there are 7 and 9 positive values for the difference of leaner knowledge and motivation, with the highest value equals to 1.43% and 16.76%, respectively. Thus, we conclude that using hybrid-CF results in a lower performance in terms of learner average knowledge. However, since hybrid-CF usually needs lower computational cost than model-based recommendation (which is not measured here) and the performance loss is not big, hence hybrid-CF is very promising in e-learning system.

Table 1. The differences between model-based and hybrid recommendation (in percentage %). The first value in each cell represents the difference of final knowledge and the second value represents the difference of final motivation.

(w_1, w_2)	FMC	MMC	SMC	NMC
(1, 0)	**0.65; 2.93**	**-0.94;** *-0.06*	**-0.76;** *-0.21*	*-0.22*; **0.00**
(U[0, 1], 0)	**1.43; 7.73**	*-0.40*; **4.21**	*-0.02*; *-0.31*	**0.66; 0.00**
(U[0, .3], 0)	*-0.54*; **14.8**	**-0.58;** *1.88*	*0.46*; **-4.61**	**1.14; 0.00**
(1, U[0, .3])	**-0.9;** *1.07*	**-1.08; -1.43**	**-0.95; -1.47**	**-0.44; 0.00**
(U[0, 1], U[0, .3])	*0.45*; **5.75**	**-0.67; 1.73**	*-0.01*; *-0.55*	**0.63; 0.00**
(U[0, .3], U[0, .3])	**-1.36; 16.76**	**-0.84;** *-0.29*	*-0.11*; **-4.53**	**0.99; 0.00**
(1, U[0, 1])	**-0.8;** *0.46*	**-1.08; -2.2**	**-0.74; -1.29**	*-0.27*; **0.00**
(U[0, 1], U[0, 1])	*0.54*; **6.74**	*-0.13*; **3.24**	*-0.06*; *-0.36*	**0.81; 0.00**
(1, 1)	**-0.97;** *1.23*	**-0.72; -1.99**	**-0.65; -1.24**	*-0.29*; **0.00**

5 Concluding Remarks

In this paper, we identified the difference of recommendation in e-learning and other areas. By using simulation, we have shown the differences between hybrid-CF and model-based recommendation in terms of the learner knowledge and motivation. However, our simulation is only limited to the adjustment of some parameters. In the future, we will change some other parameters, and provide a more detailed experiment, both by increasing the complexity of the artificial learner and by changing the recommendation criteria.

References

1. Basu, C., Hirsh, H., Cohen, W. and Nevill-Manning, C. 2001. Technical Paper Recommendations: A Study in Combining Multiple Information Sources. *JAIR*, 1, 231-252.
2. McNee, S.,Albert, I.,Cosley, D.Gopalkrishnan, P.,Lam, S., Rashid, A., Konstan, J. and Riedl, J. 2002. On the Recommending of Citations for Research Papers. *ACM CSCW'02*. 116-125.
3. Tang, T.Y. and McCalla, G. 2003. Towards Pedagogy-Oriented Paper Recommendation and Adaptive Annotations for a Web-Based Learning System. In *IJCAI 2003 Workshop on Knowledge Representation and Automated Reasoning for E-Learning Systems*, Acapulco, Mexico. 72-80.
4. Woodruff, A., Gossweiler, R., Pitkow, J., Chi, E. and Card, S. 2000. Enhancing a Digital Book with a Reading Recommender. *ACM CHI 2000*, 153-160.

Generation of Demand Feedback in Intelligent Tutors for Programming

Amruth N. Kumar

Ramapo College of New Jersey, Mahwah NJ 07430, USA
amruth@ramapo.edu

Abstract. In this paper, we will examine how a model-based tutor can automatically generate demand feedback. We will propose a two-stage feedback generation algorithm that maintains the principle of modularity characteristic of model-based representation, while automatically generating detailed demand feedback. We will evaluate model-based programming tutors to demonstrate that the feedback generated using this algorithm is coherent and effective at improving learning.

1 Introduction

We have been developing intelligent tutors to help students learn specific constructs in programming languages, such as loops, pointers, and expression evaluation. The tutors present a code segment involving a construct, ask the learner to solve problems based on the code, and provide demand feedback to facilitate learning. The tutors generate the problems as parameterized instances of problem templates (as in [6]).

Various technologies have been used to model the domain in an intelligent tutor, including rule-based reasoning (e.g., production rules used in ACT-R theory [1]), constraint-based reasoning (e.g., [10]), and case-based reasoning (e.g., [13]). Earlier, we had proposed using model-based reasoning for programming tutors [7] because of its relative advantages over the other technologies [5]. In this paper, we will briefly examine how demand feedback can be generated in a tutor that uses model-based reasoning for domain modeling.

One of the advantages of using model-based reasoning for modeling the domain in an intelligent tutor is that the domain model doubles as the runnable expert module [7]. Therefore, the tutor is capable of solving problems on its own. Can the tutor also **automatically generate feedback** on its own?

The representation used in model-based reasoning is modular - each component is responsible for only its behavior. A device derives its behavior by composing the behaviors of its components. Such a modular design promotes scalability. Can such a modular scheme be used to generate feedback? If so, would the resulting feedback be **coherent**? Would it be **effective** enough for the user to learn from? In this paper, we will briefly examine these questions.

2 Feedback Generation

Currently, our tutors provide demand feedback [2], i.e., they provide feedback only when the learner demands. Research shows that students who practice with demand feedback are significantly better at far-transfer tasks than those who receive immediate feedback [4][9]. Therefore, demand feedback is appropriate for our tutors on programming.

Our tutors generate demand feedback using simulation and reflection. Reflection is the process of providing explanation by reflecting on the state and the knowledge of the tutor [11]. During simulation, each tutor executes the model of the program. By way of reflection, it examines the state of the components, and generates explanation based on its observations. The resulting feedback can not only explain the behavior of the program, but also justify the correct answer.

We use a two-stage algorithm to generate feedback about a program, by composing the feedback generated by its components:

- **Process Explanation:** The interpreter that executes the code generates this explanation in a fashion similar to program tracing. For each line of code, the interpreter identifies the components participating in the line of code, and any state transitions that they undergo as a result of executing the line of code. This includes identifying any side-effect that results from executing the line, i.e., any input, output, or change in the values of variables. Since the lines of code are executed in the order specified by the program, the resulting feedback narrative is coherent.
- **Component Explanation:** This explanation is generated by the components participating in the line of code that is currently being executed. If an action is applied to a component that is not supported by the current state of the component, the component explains why the attempt to apply the action is an error. Furthermore, the component steps through and generates explanation for each of the previous states of the component. It presents the generated lines of explanation in one of two forms:
  - Detailed form: It explains all the prior states in order, and how they culminate in an error when the current action is applied.
  - Abbreviated form: It explains only the current state, and how applying the current action results in an error.

Component explanation is coordinated with process explanation as follows: during the generation of process explanation, at each line of code, component explanation is obtained from all the components participating in that line of code. Only the explanations from the components that are in an error state are inserted into the process explanation. The abbreviated form of component explanation is used during the generation of demand feedback. The algorithm may be summarized as follows:

```
Execute the code line by line:
For each line of code
   Generate process explanation
   For each component involved in the line
      If the component reached an error state on this line
         Generate abbreviated component explanation for the
                                                       component
         Insert it after the process explanation for the line
```

The tutor post-processes the feedback generated by process and component explanations to address the needs of the user:

- Detailed Feedback: Since the feedback includes a complete explanation of the behavior of the program, it is used for novices, for the first few problems in a tutoring session, and in instructional (as opposed to problem-solving) mode.
- Diagnostic Feedback: The feedback is post-processed to include only the following:
  - Process explanation generated by input and output statements;
  - The abbreviated version of component explanation generated by components with observed errors.

  Diagnostic feedback is used once the student starts making progress towards the educational objectives of the tutor.

The algorithm can be applied to any domain wherein the composition of behavior reflects the composition of structure, i.e., those and only those components that constitute a device contribute to the behavior of the device.

2.1 Example

The following is a typical program that the learner is asked to debug when using a tutor on pointers:

```
void main()
{
   long *indirectPointer;
   {
      long count = 1313;
      indirectPointer = &count;
   }
   cout << *indirectPointer;
}
```

The demand feedback generated by the tutor for the above code is:

```
When the program is executed:
The program always starts executing from main().
When the function main() is executed:
Pointer indirectPointer is declared on line 4.
But, it has not been assigned a value yet
```

```
When the nested block starting at line 5 is executed:
Variable count is declared on line 6.
It is initialized during declaration to 1313
Pointer indirectPointer is assigned to point to count on line 7
The nested block is exited on line 8
Variable count goes out of scope and is deallocated on line 8
An attempt is made to print the value of the variable pointed to
by indirectPointer on line 9
```
But, variable count has already gone out of the scope of its declaration.
Therefore, indirectPointer is a dangling pointer.
```
The function main() is exited on line 10
Pointer indirectPointer goes out of scope and is deallocated on
line 10
```

All except the italicized lines of feedback are generated during process explanation. The italicized lines are an abbreviated version of component explanation. Note that it is important for component explanation to include the complete context so that its feedback can be understood independent of any other feedback generated by process explanation.

3 Evaluation of the Tutor

We have evaluated several of our tutors to check if demand feedback generated automatically by the tutor is effective enough to promote learning. In each case, we used a pre-test, practice, post-test protocol. The pretest and post-test were of comparable hardness, with similar problems, listed in the same order. We used controlled tests - during the practice, the control group was provided minimal feedback and the experimental group was provided detailed demand feedback. Minimal feedback listed whether the learner's answer was correct or wrong, but not why. Detailed demand feedback also explained why, and was automatically generated by the tutor.

Where possible, in order to eliminate practice effect, we considered student score per attempted question or percentage of attempted questions correctly answered, instead of raw scores. We calculated effect size as (post-test score - pretest-score) / standard-deviation on the pre-test. Note that an effect size of two is the holy grail for tutors - one-on-one human tutoring is known to improve learning by two standard deviations over traditional classroom instruction [3]. We used 2-tailed p-value to verify the statistical significance of our results.

When we evaluated our tutor on expression evaluation tutor in Fall 2002, we found that the effect size was 1.35 for the test group (N=24) that received detailed feedback versus 0.84 for the control group (N=24) that received minimal feedback. Similarly, when we evaluated our tutor on parameter passing in Spring and Fall 2002, we found that the effect size was 0.99 for the test group (N=14) that received detailed feedback versus 0.07 for the control group (N=15) that received minimal feedback. These and other similar results (e.g., [8]) indicate that the demand feedback automatically generated by our tutors using the two-stage algorithm described in Section 2 is coherent and effective in promoting learning. We plan to continue to evaluate our tutors for the quality and quantity of feedback they provide.

Acknowledgments. Partial support for this work was provided by the National Science Foundation's Course, Curriculum and Laboratory Improvement Program under grant DUE-0088864.

References

1. Anderson, J.R.: Production Systems and the ACT-R Theory. In Rules of the Mind. Hillsdale, NJ: Lawrence Erlbaum & Associates, Inc. (1993) 1–10.
2. Anderson J.R., Corbett A.T., Koedinger K.R. and Pelletier R.: Cognitive Tutors: Lessons Learned. The Journal of the Learning Sciences, Lawrence Erlbaum Associates, Inc. Vol No 4(2) (1995) 167–207.
3. Bloom, B.S.: The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring. Educational Researcher, Vol 13 (1984) 3–16.
4. Corbett, A.T. and Anderson, J.R.: Locus of feedback control in computer-based tutoring: impact on learning rate, achievement and attitudes. Proceedings of CHI 2001. ACM 2001. (2001) 245–252.
5. Davis, R.: Diagnostic Reasoning Based on Structure and Behavior. Artificial Intelligence, 24 (1984) 347–410.
6. Koffman, E.B. and Perry, J.M.: A Model for Generative CAI and Concept Selection. International Journal of Man Machine Studies. 8 (1976) 397-410.
7. Kumar, A.N.: Model-Based Reasoning for Domain Modeling in a Web-Based Intelligent Tutoring System to Help Students Learn to Debug C++ Programs. Proceedings of Intelligent Tutoring Systems (ITS 2002), LNCS 2363, Biarritz, France, June 5-8, 2002, 792–801.
8. Kumar, A.N.: Learning Programming by Solving Problems, Informatics Curricula and Teaching Methods, L. Cassel and R.A. Reis ed., Kluwer Academic Publishers, Norwell, MA, 2003, 29–39.
9. Lee, A.Y.: Using tutoring systems to study learning. Behavior Research Methods, Instruments and Computers, 24(2), (1992) 205–212.
10. Martin, B. and Mitrovic, A.: Tailoring Feedback by Correcting Student Answers. Proceedings of Intelligent Tutoring Systems (ITS) 2000. G. Gauthier, C. Frasson and K. VanLehn (eds.). Springer (2000) 383-392.
11. Neuper, W.A.: A 'calculemus-approach' to high-school math? In S. Linto and R. Sebastiani (eds.), Proceedings of the 9th Symposium on the Integration of Symbolic Computation and Mechanized Reasoning, Siena, Italy, (June 2001).
12. Reiser, B., Anderson, J. and Farrell, R.: Dynamic student modeling in an intelligent tutor for LISP programming, in Proceedings of the Ninth International Joint Conference on Artificial Intelligence, A. Joshi (Ed.), Los Altos CA (1985).
13. Reyes, R.L. and Sison, R.: A Case-Based Reasoning Approach to an Internet Agent-Based Tutoring System. In: Moore, J.D. Redfield, C.L. and Johnson, W.L. (eds.): Artificial Intelligence in Education: AI-ED in the Wired and Wireless Future, IOS Press, Amsterdam (2001) 122–129.

Using Language to Determine Success in Negotiations: A Preliminary Study*

Marina Sokolova, Stan Szpakowicz, and Vivi Nastase

School of Information Technology and Engineering
University of Ottawa, Ottawa, Canada

Abstract. Negotiation support systems often allow an exchange of messages that help explain better the offers and positions of the negotiators. Collections of such messages can be analyzed using Natural Language Processing techniques. We work with a large collection accumulated by the Inspire system. The messages are unedited and share characteristics with email text data. We use them to classify negotiations as successful or failed, and to find language patterns characteristic of these two classes. The preliminary results show that certain patterns in the language of the messages do predict whether the negotiation will succeed.

1 Texts in Electronic Negotiations

We investigate how language reflects success or failure of electronic negotiations (e-negotiations). We aim to build a language model of texts that accompany fixed-problem negotiations performed via a negotiation support system (NSS). Language models of e-negotiations can help understand how the negotiators' behaviour reflects their strategy and tactics, and balance the negotiators' subjective self-evaluation with an external, and presumably objective, source of evaluation. We seek models that relate success or failure of negotiations to such text characteristics as word statistics and lexical semantics. We report text characteristics discovered as intermediate results in building such a model. This appears to be the first study of e-negotiation textual data that uses Natural Language Processing (NLP) techniques, although other research communities actively investigate such data and the underlying processes [3,2].

In e-negotiations people do not have visual or acoustic information to evaluate the negotiation process, plan their future actions and further their goals. In free-form *written* messages exchanged, language is essential for understanding the process and outcome of negotiations. To identify language patterns indicative of success or failure of negotiations, we apply statistical analysis to 1482 negotiation texts collected by the NSS Inspire [3]. Inspire is a research and teaching tool used in universities and colleges in more than ten countries. There were over 2500 contributors of texts mostly written in English.

Inspire is a support environment for bilateral negotiations. Every negotiation has a buyer ("Cypress Cycles") and a seller ("Itex Manufacturing") who seek

* Work supported by a major SSHRC grant and an NSERC doctoral scholarship.

agreement on trading bicycle parts. Negotiation issues are fixed (part price, delivery schedule, payment terms, policy on returns) and have pre-specified sets of values, so there can be exactly 180 different offers. Inspire also mediates information exchange during the negotiation: compulsory numerical tables represent offers, optional messages complement offers or are exchanged between offers. Message texts are informal and poorly edited [6]. In this study we keep all orthographic, spelling and grammatical errors. The results show that, while the negotiators' negotiation skills, educational and cultural background and English skills vary, their language shows general trends.

2 Identifying Negotiation Success in Texts

Aiming to determine success from the negotiators' language, we are interested in text data that provide substantial information about negotiations. If messages were absent or were not in English, we ignored the negotiation. This left 1275 negotiations, 761 of them successful. There are 809,584 word tokens and 20,240 word types. This differs from the Brown corpus with 1 million tokens and 53,850 types. We attribute the high token-type ratio to the fixed topic of discussion.

Inspire users discuss a fixed topic, negotiate a fixed problem, and choose among fixed options. We want to know how these conditions affect the growth of the vocabulary and its convergence with respect to the sample size [4]. We add information on rare words based on *hapax legomena* ($V(1, N)$) and *dis legomena* ($V(2, N)$) [9]; $V(i, N)$ is the number of types that occur i times in the text with N tokens. We calculate the growth rate of the vocabulary $P(N) = V(1, N)/N$ and Sichel's characteristic $S(N) = V(2, N)/N$. Table 1 shows that new words are steadily added at every stage; $TT(N)$ is the type-token ratio. Moreover, the vocabulary grows approximately at the same rate through the data, and Sichel's characteristic converges as expected [9]. That is, the Inspire vocabulary grows as the vocabulary of unrestricted languages [4].

Table 1. The lexicon growth rate of the Inspire data.

N	96940	293152	364306	535244	614428	716176	809584
$TT(N)$	0.060	0.040	0.033	0.029	0.027	0.026	0.025
$P(N)$	0.027	0.018	0.015	0.013	0.012	0.011	0.010
$S(N)$	0.008	0.005	0.004	0.0037	0.0035	0.0033	0.0031

For further studies we concatenate all messages that accompany a negotiation and treat it as one entry. This is justified by the nature of bilateral negotiation: in a dialogue, a message depends on or relates to the previous messages. We focus on finding language patterns representative of successful (failed) negotiations, that at the same time cannot represent failed (successful) negotiations. For example, *I can accept* appears almost 3 times more often in successful than in failed negotiations; *you accept my* appears twice as often in failed than in successful negotiations. That is, we look for features frequent in successful and rare or absent in failed negotiation – or conversely. To find such features, we separate the two classes: we construct and investigate two data sets. For both, we build lists of unigrams, bigrams and trigrams, and rank the N-grams by frequency [1]. We compare the ranks of the same N-gram in successful and failed negotiation data, and find N-grams frequently present in one set and rarely present or absent in the other. These N-grams are easily detected when we plot N-grams from successful and failed negotiations; see Fig. 1. The N-grams with a large difference in ranks are depicted as the outliers. Note that the graph for 761 successful negotiations lies, predictably, above the graph for 514 failed negotiations.

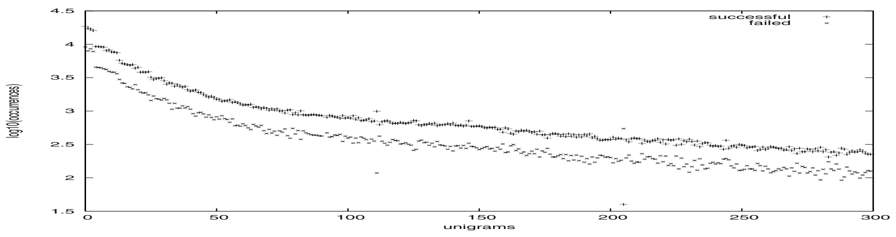


Fig. 1. Successful and failed negotiations

Previous studies on classifying e-negotiations did not consider the language aspect of negotiations [3]. We state our **first hypothesis**: the use of negotiation-related words is relevant to the success of negotiations. To confirm, we run experiments to classify negotiation as successful or failed based on the use of 123 most frequent negotiation-related words. For each negotiation we build a bag of those words [1], counting their occurrences. We add the count of the remaining words to the bag, so each negotiation is represented by a vector of 124 integer-valued features and a binary-valued feature for success/failure. We employ several Machine Learning(ML) classifiers freely available in the Weka suite [10] – AD Trees (ADT), Decision Stumps (DS), Decision Tables (DT), Instance-based using 20-nearest neighbour (IBK), analog of Support Vector Machine (SMO) – and the Decision List Machine (DLM) [5]. We use 10-fold cross-validation. Our experiments give 66-69% overall classification accuracy with the baseline (BL) 60%. Precision and recall are calculated with respect to successful negotiations. See Table 2.

Table 2. Classification of negotiations.

Measure	BL	ADT	DLM	DS	DT	IBK	SMO
Precision	60 %	67.4 %	67.2 %	68.2 %	68.1 %	70.3 %	72.9 %
Recall	100 %	84.2 %	90.5 %	83 %	85.2 %	70.4 %	72.3 %

Simons [8] found that language patterns of the first part of negotiation efficiently predict the negotiation outcome. In our data both company names, *Cypress* and *Itex*, have higher ranks in failed than in successful negotiations. We note that company names mostly appear when the negotiators introduce themselves. The rank difference suggests that the language of successful and failed negotiations differ from the very beginning of the process. Our **second hypothesis**: the starting phases of successful and failed e-negotiations differ.

In a preliminary evaluation of this hypothesis, we find that in successful negotiations *Cypress* and *Itex* together account for 3/4 of the data they account for in failed negotiations. We also compare ranks of frequent bigrams and trigrams containing company names among 500 most frequent bigrams and 700 most frequent trigrams. Their ranks on the list for successful negotiations are r_s, for failed negotiations – r_f. *Cypress* appears in successful negotiations in one bigram ($r_s = 96$) and two trigrams ($r_s = 467, 532$) with 0.0555 % of text covered. In failed negotiations *Cypress* appears in one bigram (r_f 55) and five trigrams ($r_f = 359, 391, 420, 474, 625$) with 0.1495 % of text covered. *Itex* appears in successful negotiations in one bigram ($r_s = 285$) and no trigrams. In failed negotiations *Itex* appears in one bigram ($r_f = 295$) and no trigrams. The comparison of the N-gram ranks suggests that both in successful and failed negotiations the buyer's name is more frequent than the seller's name. *Cypress* is noticeably more frequent in failed negotiations. *Itex* is more frequent in successful negotiations.

Our **third hypothesis** is that politeness characterizes successful negotiations. To support it, we find the unigram ranks of such indicators of polite speech as *Thank* ($r_s = 69$, $r_f = 83$), *Thanks* ($r_s = 108$, $r_f = 109$), *thank* ($r_s = 250$, $r_f = 315$), *thanks* ($r_s = 420$, r_f 379). We deliberately preserve case-sensitivity, to enable further studies of negotiators' attitude towards the negotiation process and negotiation partners. The combined percentage of the orthographic variations of the words *thank* and *thanks* is 2.5 times higher in successful than in failed negotiations.

We employ bigrams and trigrams to find more language differences between successful and failed negotiations. We are looking for trigrams that show the negotiators' goal (win by any means, reach a compromise, do away with the assignment), their attitude to partners (friendliness, aggressiveness, indifference), and behaviour in the negotiation process (flexibility, stubbornness). The same trigrams must be noticeably present in either successful or failed negotiations. We notice that in the trigrams from the failed negotiations there is a trace of aggressive behaviour (**you will accept, you will agree, you are not**), which is absent from the corresponding trigrams in the successful negotiations (**you can accept, agree with your, it is not**). Tracing the trigrams with "you", we found that in successful negotiations they correspond to politeness, in failed negotiations – to aggressiveness. See [7] for more on the methods used and language patterns found.

3 Conclusions and Future Work

We presented a procedure for identifying language patterns indicative of the outcome of negotiations. Using NLP and ML techniques, we showed how text messages exchanged by the users of an NSS are relevant to the outcome of e-negotiations. We related language with the success of negotiations by arguing in favour of three hypotheses. We showed that e-negotiations can be classified correctly if they are represented by bags of words built from negotiation-related words. We showed some ways in which language differs in successful and failed negotiations, in particular in initial phases. Also, politeness is a factor in successful negotiations.

We did not emphasize differences between the language patterns of buyers and sellers. Role-dependent patterns are a promising direction of ongoing work. We will use ML methods to find language patterns. How ML methods are implemented poses questions about different mappings of negotiations to bags of words. This is closely linked to the question how the use of words not related to negotiation relates to negotiation outcomes. Finally, there is the issue of how noise in text affects the e-negotiation process, also left for future work. Preliminary experiments show correlation between text data not related to negotiation, noise level and the negotiation outcomes.

References

1. D. Jurafsky and J. H.Martin. *Speech and Language Processing.* Prentice Hall (2000).
2. G. E. Kersten. "The Science and Engineering of E-negotiation: An Introduction". InterNeg Report 02/03, 2003. interneg.org/interneg/research/papers/
3. G. E. Kersten, G. Zhang. "Mining Inspire Data for the Determinants of Successful Internet Negotiations", *Central European J. of Operational Research*, **11**(3), 297-316, 2003.
4. M. P. Oakes. *Statistics for Corpus Linguistics.* Edinburg University Press, 1998.
5. M. Sokolova, M. Marchand, N. Japkowicz, J. Shawe-Taylor "The Decision List Machine", *Advances in Neural Information Processing Systems*, **15**, 921-928, The MIT Press, 2003.
6. M. Sokolova, S. Szpakowicz, V. Nastase. "Automatically Building a Lexicon from Raw Noisy Data in a Closed Domain". INR01/04, 2004. http://interneg.org/interneg/research/papers/
7. M. Sokolova, S. Szpakowicz, V. Nastase. "Language Patterns in Text Messages of Electronic Negotiations: A Preliminary Study". INR05/04, 2004. http://interneg.org/interneg/research/papers/
8. T. Simons. "Speech Patterns and the Concept of Utility in Cognitive Maps: the Case of Integrative Bargaining". *Academy of Management J.*, **38**(1), 139-156, 1993.
9. F. J. Tweedie, R. H. Baayen. "How Variable May a Constant be? Measures of Lexical Richness in Perspective". *Computers and the Humanities*, **32**, 323-352, 1998.
10. I. Witten, E. Frank. *Data Mining*, Morgan Kaufmann, 2000. http://www.cs.waikato.ac.nz/ml/weka/

Distributed Data Mining vs. Sampling Techniques: A Comparison*

Mohamed Aounallah, Sébastien Quirion, and Guy W. Mineau

Laboratory of Computational Intelligence,
Computer Science and Software Engineering Department,
Laval University, Sainte-Foy, Québec, G1K 7P4, Canada;
{Mohamed.Aoun-Allah, Guy.Mineau}@ift.ulaval.ca, SQuirion@gel.ulaval.ca

Abstract. To address the of mining a huge volume of geographically distributed databases, we propose two approaches. The first one is to download only a sample of each database. The second option is to mine each distributed database remotely and to download the resulting models to a central site and then aggregate these models. In this paper, we present an overview of the most common sampling techniques. We then present a new technique of distributed data-mining based on rule set models, where the aggregation technique is based on a confidence coefficient associated with each rule and on very small samples from each database. Finally, we present a comparison between the best sampling techniques that we found in the literature, and our approach of model aggregation.

1 Introduction

This paper deals with the problem of mining several huge geographically distributed databases, proposing and comparing two data mining techniques. The first one that we examined uses a sampling of each individual database to which, once gathered, we apply some data mining technique. This technique is based on the aggregation of data. With this intention, we studied the existing sampling techniques. The most promising methods, based on our tests, will be detailed later in this paper.

The second technique of data mining, that we introduce (based on the aggregation of models), consists of applying data mining techniques on each individual database. The models resulting from these techniques are then gathered and some aggregated model is produced by a technique described in what follows. In this work, models, either produced individually on a subset of the data or from the aggregation technique that we propose are a set of classification rules.

This paper proceeds as follows. In Section 2, an overview of the most common sampling techniques is presented. Then, in Section 3, we present our solution to distributed data mining (DDM) using model aggregation (DDM-MA). In Section 4, we present our experimentation results. We finally present a conclusion and our future work.

* This work is sponsored by NSERC.

2 Sampling

The sampling approach consists in creating a representative sample of a large database under the hypothesis that a classifier trained from that sample will not perform significantly worse than a classifier trained on the entire database. In our context, sampling is used on each remote database, generating distinct samples at each site. We then merge these samples to finally train a classifier on the resulting sample. The literature in data mining is filled with various sampling algorithms [3] [6] [4], which can be grouped, with respect to how the samples are formed, to form three distinct families: static, dynamic and active sampling.

Static Sampling refers to a sampling that is performed without any knowledge other than what the database provides. The most common algorithm for static sampling could be called random sampling. As presented in [3], for a database D, an initial sample size n_0 and a schedule of sample size increments (typically either an arithmetic ($\Delta n_i = \lambda$) or geometric ($\Delta n_i = n_{i-1}$) schedule [6]), we first form an initial sample S of n_0 random items from D and, while the distribution of the attributes of S differs significantly from that of D, add to S an additional Δn_i random items from D \ S.

Dynamic sampling differs from static sampling only in the sample validation process. At each iteration, a classifier is built from the current sample and evaluated. If the resulting classifier has not reached satisfactory accuracy, i.e. reaches a plateau in its learning curve, the algorithm must iterate once more. There are three common techniques to detect convergence: Local Detection (LD)(stopping when $acc(n_i) \leq acc(n_{i-1})$) [3], Learning Curve Estimation (LCE) [3] and Linear Regression with Local Sampling (LRLS) [6].

Active sampling differs from dynamic sampling only in the way the items are picked at each iteration. In the literature, active sampling is used in contexts where classified items are not provided to the learner *a priori*. The learner has to pick amongst unclassified items and ask an expert for their classification. The purpose of active sampling is then to minimize the number of items needed by the learner to acquire the concept correctly. They achieve this by estimating which items would provide the greatest *knowledge gain* (formally, an effectiveness score) and including them in the sample. In general, the effectiveness scores (ES) are computed either by a probabilistic classifier or by a committee of classifiers. In our context (using classified items), the different active sampling methods can be summarized by this algorithm:

```
1. i $\Leftarrow$ 0
2. S $\Leftarrow$ {$n_0$ random items from D}
3. Generate {C}, a set of classifiers, from S
4. While {C} has not converged
   a) i $\Leftarrow$ i + 1
```

```
b) For each $x_j \in$ D \ S, compute $ES_j$, the ES, with {C}
c) S ⇐ S ∪ {$\Delta n_i$ items chosen from D \ S with respect to ES}
d) Generate {C} from S
```

Generally, the Δn_i items added to S at each iteration are the items with the highest ES. However, [8] proposes to use the ES as the item's weight in a random selection in order to add robustness to noisy data. Also interesting, [4] proposes, for a small cost in accuracy, to build the sample using a quick probabilistic classifier and then use that sample to train any other classifier. Our implementation of active sampling is an uncertainty sampling integrating these two approaches (Weighted Heterogeneous Uncertainty Sampling) for speed and robustness purposes.

3 Distributed Data Mining Using Model Aggregation

To construct our aggregated model, hereafter called the meta-classifier, we use two types of software agents: *miner agents* which mine individual (distributed) databases and *collector agents*, that are responsible for aggregating information produced by miner agents. There is typically only one collector agent in our system. Roughly speaking, our technique goes through the following algorithm. A detailed description with justifications of our choices can be found in [1].

```
1. Do, by miner agents, in parallel at different remote sites,
   for each database $DB_i$ with $i = 1 \ldots nd$, where $nd$ is the number
   of remote sites:
   a) Apply C4.5 over $DB_i$ then transform the decision tree
      obtained to a rule set $R_i = \{r_{ik} \mid k \in [1..nr_i]\}$, where $nr_i$
      is the number of rules;
   b) Compute for each $r_{ik}$ a confidence coefficient $c_{r_{ik}}$ as one
      minus the error rate of $r_{ij}$ and minus one half the width of
      the confidence interval of the error rate computed based on
      the Central Limit theorem
   c) Extract a sample $S_i$ from $DB_i$.
2. Do, by a collector agent, at a central site:
   a) Create $R$ and $S$ as follows:
      $R = \bigcup_{i=1 \ldots nd} R_i$
      $S = \bigcup_{i=1 \ldots nd} S_i$;
   b) From $R$, eliminate rules which have a confidence coefficient
      lower than a certain threshold $t$: $R_t = \{r_{ik} \in R \mid c_{r_{ik}} \geq t\}$;
   c) Create a binary relation $\mathcal{I}$ defined over $R_t \times S$ where, at
      each intersection $(r_i, s_j)$, we find $0$ if $r_i$ does not cover
      $s_j$, $1$ if $r_i$ covers $s_j$ correctly, or $-1$ otherwise ;
   d) For each rule $r \in R_t$, compute an error rate $Err_{\mathcal{I}}^{r}$ using $S$ as
      test set, i.e. the number of $-1$ in each row of $\mathcal{I}$ divided by
      the number of non-zero values in the same row;
   e) Construct the rule set $R_{t_{\mathcal{I}}}$ using a threshold $t_{\mathcal{I}}$ as follows:
      $R_{t_{\mathcal{I}}} = \{r_p \in R_t | Err_{\mathcal{I}}^{r_p} \leq t_{\mathcal{I}}\}$.
```

4 A Comparative Experimentation

For our experiments, we have tested techniques proposed on nine data sets: adult, chess end-game (King+Rook versus King+Pawn), house-votes-84, ionosphere, mushroom, pima-indians-diabetes, tic-tac-toe, Wisconsin Breast Cancer (BCW)[5] and Wisconsin Diagnostic Breast Cancer (WDBC), taken from the UCI repository [2]. The size of these data sets varies from 351 to 45222 objects.

In order to determine the best sampling techniques, we divided each database into a training set (2/3) and a test set (1/3), when they are not already divided in the UCI repository. On the other hand, to test the DDM technique proposed and in order to simulate a distributed environment, firstly, we divided each database into two data subsets with a proportion of 1/4 and 3/4. The first subset is used as test set for the meta-classifier (aggregated model) or for the classifier built on the samples aggregation. The second subset is randomly divided into two, three or four data subsets of random sizes, which are, in turn, each divided into two sets with proportion of 2/3 (a data file) and 1/3 (the associated .test file) for training and test sets respectively. For the meta-classifier, a random sample set (an associated .sple file) is extracted from the training set (the .data file) with a size of 10% its size and a maximum of 50 objects. This maximum size is needed to bound the meta-classifier technique to very small data samples, which is in accordance to one of our assumptions.

4.1 Comparing the Sampling Methods

In order to compare sampling followed by a data mining technique on the aggregated samples, with distributed data mining as proposed in this paper, we decided to compare the various sampling methods to determine the one that succeeds best on our test data. So we compared both methods, dynamic and active sampling, using each of the three convergence detection mentioned earlier (*LD*, *LCE* and *LRLS*), testing each of these six methods with an arithmetic (*Arith.*) and geometric (*Geo.*) schedule. We also compared these methods to random sampling, with an arithmetic and geometric schedule, and to samples of 50 items formed by random picking and by weighted uncertainty sampling, for a total of 16 competing sampling methods.

Experiments have shown that "Active - LCE - Geo" and "Dynamic - LCE - Geo." are almost always among the three best methods. For each data set where this is not the case (i.e., one of these two methods does not appear among the best three methods), experiments have also shown that the methods "Active/Dynamic - LCE - Geo." are always within 5% of the most accurate method on any of the nine databases. These results suggest that these are the two most effective sampling techniques, at least for data sets that would resemble ours.

4.2 The DDM-MA Experiment

For the construction of the base classifiers we used C4.5 release 8 [7] which produces a decision tree that is then directly transformed into a set of rules. The confidence coefficient of each rule is computed on the basis of 95% confidence interval (i.e., $N = 95$). For threshold $t_{\mathcal{I}}$, we used 5% and 10% respectively, but these two values gave exactly the same results. For threshold t we used i) all values ranging from 0.95 to 0.20, with decrements of 0.05, ii) 0.01, iii) and, μ with $\mu = 1/nd \sum_{i=1}^{nd} \mu_i$ and $\mu_i = 1/nr_i \sum_{k=1}^{nr_i} c_{r_{ik}}$. The value μ is used in order to get an automatic way to compute the threshold based on the average confidence that we have in the produced rules.

Experimentations of the meta-classifier with the different values of threshold t, previously listed, showed that μ gave the best performance. This is predictable since it is not an absolute value but rather it is a threshold that finds a consensus between the different μ_i by finding their closest value.

4.3 Comparison between Meta-classifier and Sampling

We base our comparison on the results obtained in sections 4.1 and 4.2. Consequently, in this section, we only compare Dynamic/Active - LCE - Geo. sampling techniques with the meta-classifier with $N = 95$, threshold $t = \mu$ and threshold $t_{\mathcal{I}} = 5\%$. Comparison is conducted on the basis of error rate and execution time. In order to assess the importance of the error rates obtained by the meta-classifier and sampling techniques, we compare them to error rates obtained on C4.5 applied to the whole database $DB = \cup_i DB_i$; it is used only as a reference to assess the loss accuracy since, by assumption, we stated that we could not process DB because of download/processing time constraints.

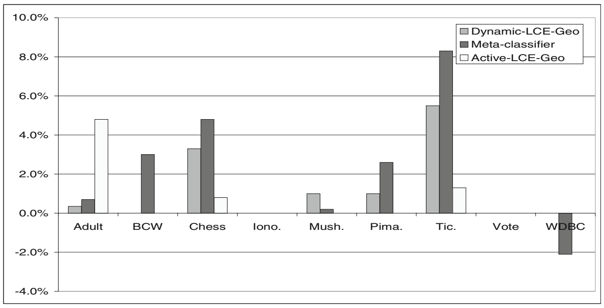


Fig. 1. Error rate comparison between the meta-classifier and the Active/Dynamic LCE Geo. sampling techniques assuming C4.5 error rate as reference

Figure 1 shows the difference between what we obtained using C4.5, versus the sampling techniques and the meta-classifier. The Dynamic - LCE - Geo sampling technique is represented by a light gray histogram, the Active - LCE - Geo sampling technique is represented by a white histogram and the meta-classifier approach is represented by a dark gray histogram. The first conclusion that we can extract from this chart is that all these error rates could be assessed to be acceptable since they are no more than 8.5% worse than C4.5 performance.

Comparing sizes. comparing the the size of each database to on one hand the size of the samples obtained with Active/Dynamic-LCE-Geo. sampling techniques and on the other hand the meta-classifier[1], we can conclude that in 4 cases (BCW, Iono., Vote and WDBC) the size of the samples issued from Active/Dynamic-LCE-Geo. sampling is the same as the database. This explains the fact that in Fig. 1 the error rate of the sampling techniques is the same as that of the C4.5 algorithm for these 4 cases producing a difference of 0. Surprisingly, in these 4 data sets, our meta-classifier gives the same error rate as the C4.5 in two cases (Iono. and Vote), 3% worse (BCW) or even better (WDBC) with samples as small as 34 items or less. In the 5 other cases, our meta-classifier has an error rate comparable to sampling techniques: better than one of the two techniques or worse than the worse sampling technique by no more than 2.80%. This performance is quite interesting since the meta-classifier sample sizes are much smaller than those required by the sampling techniques.

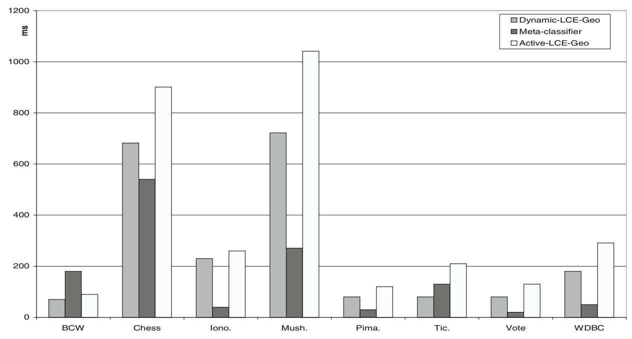


Fig. 2. Execution time comparison between the meta-classifier and the Active/Dynamic LCE Geo. sampling techniques

Comparing processing time. Finally, in order to compare the sampling techniques with the DDM-MA method on the basis of execution time, we can look at Fig. 2. Noting that these programs were programmed in C++, compiled by the same compiler and executed (single task dedicated CPU) on the same machine. While not presented on the chart for readability reasons, the execution times of the Dynamic/Active-LCE-Geo. sampling and the meta-classifier on the adult data set are respectively 20078ms, 38905ms and 9852ms. From this, one can easily conclude that, apart from the BCW and TIC data sets, that DDM-MA is always faster and sometimes, much faster than sampling techniques. Furthermore, the asymptotic analysis of the algorithm given in Fig. 2 shows that this would remain the case as N, the size of DB, grows.

[1] As reminder, the samples S_i used to produce the meta-classifier have a maximal limit of 50 items, but can be less if the size of DB_i is less than 500.

5 Conclusion

In this paper, we presented an overview of the most common sampling techniques as well as a new technique for distributed data mining based on the aggregation of models. Experiments highlighted the constantly high accuracy obtained by Active/Dynamic LCE Geo. sampling.

Our tests also provide us with the best parameters for the meta-classifier, which are used in our comparison with the most promising sampling techniques. Experiments showed that meta-classifier error rates are quite acceptable compared to those of sampling techniques or to that of a C4.5 applied on the whole database. Moreover, the meta-classifier uses samples of very small size compared to those produced by sampling techniques. A comparison of the processing time, showed that the meta-classifier is significantly more efficient than sampling techniques as data set size becomes important.

We can conclude that the meta-classifier presented in this paper is a promising technique. We are currently considering different approaches to improve its performance. For example, an importance coefficient could be affected to each object in a sample since the distributed databases could have significant differences in size. Moreover, we plan on integrating some efficient sampling techniques to extract the samples S_i used in the production of $\mathcal{I}$. Their impact on the various model aggregation techniques will be carefully assessed.

References

1. Mohamed Aounallah and Guy Mineau. Rule Confidence Produced From Disjoint Databases: a Statistically Sound Way to Regroup Rules Sets. *Accepted in IADIS international conference, Applied Computing.* 2004
2. C.L. Blake and C.J. Merz. UCI Repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html, 1998
3. George John and Pat Langley. Static Versus Dynamic Sampling for Data Mining. In Evangelos Simoudis and Jiawei Han and Usama M. Fayya, editors, *Proceedings of the Second International Conference on Knowledge Discovery in Databases and Data Mining*, pages 367–370, Portland, Oregon, August 1996. AAAI/MIT Press.
4. David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 3–12, Dublin, IE, 1994. Springer Verlag, Heidelberg, DE.
5. O.L. Mangasarian and W.H. Wolberg. Cancer diagnosis via linear programming. *SIAM News*, 23(5):1–18, September 1990.
6. Foster Provost, David Jensen, and Tim Oates. Efficient Progressive Sampling. In Surajit Chaudhuri and David Madigan editors, *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 23–32, N.Y., August 15–18 1999.ACM Press.
7. J. Ross Quinlan. *Improved Use of Continuous Attributes in C4.5.* Journal of Artificial Intelligence Research, 4:77–90, 1996.
8. Maytal Saar-Tsechansky and Foster Provost. Active Sampling for Class Probability Estimation and Ranking. http://www.mccombs.utexas.edu/faculty/Maytal.Saar-Tsechansky/home/MLJ-BootstrapLV-final.pdf, 2001.

Binary Decision Tree Using Genetic Algorithm for Recognizing Defect Patterns of Cold Mill Strip

Kyoung Min Kim[1,4], Joong Jo Park[2], Myung Hyun Song[3], In Cheol Kim[1], and Ching Y. Suen[1]

[1] Centre for Pattern Recognition and Machine Intelligence (CENPARMI), Concordia University, 1455 de Maisonneuve Blvd. West, Suite GM606, Montreal, Canada H3G 1M8
{kkm, kiminc, suen}@cenparmi.concordia.ca
[2] Department of Control and Instrumentation Engineering, Gyeongsang National University, 900, Gazwa-dong, Chinju, Gyeongnam, 660-701, Korea
[3] Department of Electric Control Engineering, Sunchon National University, 540-742, Korea
[4] Department of Electrical Engineering, Yosu National University, 550-749, Korea

Abstract. This paper presents a method to recognize the various defect patterns of a cold mill strip using a binary decision tree constructed by genetic algorithm(GA). In this paper, GA was used to select a subset of the suitable features at each node in the binary decision tree. The feature subset with maximum fitness is chosen and the patterns are divided into two classes using a linear decision function. In this way, the classifier using the binary decision tree can be constructed automatically, and the final recognizer is implemented by a neural network trained by standard patterns at each node. Experimental results are given to demonstrate the usefulness of the proposed scheme.

1 Introduction

To produce a cold mill strip of high quality, it is important to detect the defects on the surface of cold mill strip exactly in the manufacturing process. So, efficient methods for the recognition and extraction of defect patterns of cold mill strips have been studied [1, 8, 10].

The conventional method to recognize the defect patterns is to extract good features experimentally from the cold mill strip image acquired from a CCD camera and then recognize the patterns in a single step by inputting all the features to a neural network. But this method has two problems when the characteristics of the defect patterns are considered. Firstly, because the shapes of the defect patterns are complex and irregular, the recognition rate of defect patterns is sensitive to the kinds of selected features. And also despite the good separability of the features, they may interfere with each other when used together. Secondly, because there exist some similar classes of defect patterns, which can be classified into the same group, classifying all the patterns in only a single step results in a high classification error.

To overcome these problems, we propose a multi-stage classifier like a decision tree, which repeats decisions so as to classify patterns individually. The decision tree classifier makes fast and exact decisions by dividing the complex and global decisions into several simple and local decisions [5]. For an efficient and accurate classification, an optimal or near-optimal feature subset within the feature space needs to be selected at each decision node [2].

This paper introduces the binary decision tree and describes methods of both generating a linear decision function and selecting a feature subset using GA. Then, an automatic method of constructing the binary decision tree is described. And finally, the proposed classifier is applied to recognize the defect patterns of a cold mill strip.

2 Construction of Binary Decision Tree Using GA

In the case of a one-stage classifier, it cannot be guaranteed that all the features used in classification have the best separability, and the efficiency of the classifier is low because it compares a pattern with all of the other classes [6, 7]. To solve these problems, we have designed a classifier that decides the class of a given input pattern by repeating two or more decisions successively. This classifier is called a multi-stage classifier or a decision tree classifier.

If only the necessary features are used at each node of binary decision tree classifier, both the accuracy and the reliability of the classifier increase. So the problem is to select a valid feature subset from the entire feature set, i.e. the feature selection problem. To select the optimized feature subset, the separability of all the combinations of the features should be evaluated. However, when *m* components are selected among *n* components, the number of combinations expressed as nCm becomes a large value even if *n* and *m* are not large. The feature selection problem can be regarded as an optimization problem. So in this paper, feature selection is executed using GA, a global optimization technique.

GA has a higher probability of finding the global optimized solution than other classical optimization algorithms because it searches for the multiple global solutions simultaneously [1, 3, 4].

Fig. 1 shows the process of selecting the optimal feature subset by GA In this paper, a feature subset is evaluated by the classification error when classifying patterns with the linear decision function that is also generated by GA.

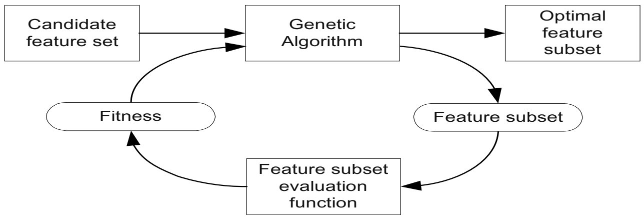


Fig. 1. Processing block diagram of feature selection

The following method is used to minimize the classification error using GA: Suppose that the given data set is $X = \{\mathbf{x}_{1,}\mathbf{x}_2, ..., \mathbf{x}_N\}$ ($\mathbf{x}_k \in R^n$ is the number of features), and *l*(*j*) and *r*(*j*) are defined as the minimum and maximum values of the *j*-th feature.

$$l(j) = \min_i \; x_{ij} \qquad (1)$$
$$r(j) = \max_i \; x_{ij}$$

In the case of 2-dimensional space, *j* can have the value of 1 or 2 and on the basis of $l(j)$ and $r(j)$, a rectangle can be constructed that can include all data. Inside the rectangle, two points can be selected arbitrarily, connected by a line. From the coefficients of the line function, a n -dimensional decision function can be obtained as follows.

$$d(\mathbf{x}) = w_1x_1 + w_2x_2 + ... + w_nx_n + w_{n+1} = \mathbf{w}_0{'}\mathbf{x} + w_{n+1} \qquad (2)$$

When matching this concept with a binary string of GA, ***n*** segments of a binary string indicate one point in the *n*-dimensional space. In the *n*-dimensional case, ***n*** points should be selected in such a way that a string is composed of n^2 segments. Supposing that the length of each segment is *m*-bit, then the total length of the binary string becomes n^2m -bits. The decision function described above is the linear decision function that minimizes the error that occurs when classifying standard patterns into two classes in a certain feature space. However, the real input patterns are not restricted to two classes of patterns. GA determines the decision function that minimizes classification error in a given feature space. Since the minimized error varies with the combination of features, the fitness function is constructed to give high fitness for a combination with a small classification error and low fitness for a combination with a large classification error.

Using the method described above, a certain feature subset minimizing classification error is chosen. And patterns are classified into two groups at each node with this feature subset. The binary decision tree is constructed by iterating this process until all the classes of patterns appear independently at each leaf node. Because the binary decision tree is constructed for multiple classes rather than just for two it is better to maintain uniform distribution for two separated groups at each node, which means it is better that two separated groups have similar numbers of classes without partiality.

To quantify this, a balance coefficient is defined using the mean and deviation of classes of a new group, as Eq. (3). If the number of patterns of the two separated groups is similar, the balance coefficients are smaller. In this case, because the depth of the binary tree becomes small, the matching time required for recognizing a pattern decreases.

$$balance = \sqrt{\frac{\sum_{j=1}^{h}(N_j - \frac{N}{h})^2}{(\frac{N}{h})^2}} \qquad (3)$$

In Eq. (3), *h* is the number of nodes, *N* is the number of input patterns, and N_j is the number of the patterns included in the *j*-th node. In this paper, a binary tree is constructed, so *h* becomes ***2***. The fitness function that includes the balance coefficient is defined as.

$$fitness = \frac{1}{1 + w_e \cdot error + w_b \cdot balance} \tag{4}$$

In Eq. (4), *error* and *balance* are the classification error and the balance coefficient between groups, respectively, and w_e and w_b are the weights for weighting each parameter. And the result of the constructed tree can be varied by adjusting of the weights w_e and w_b. After the construction of the binary decision tree, by training BP neural network with the feature subset selected optimally at each node, the final binary tree structured recognizer is realized.

3 Classification of the Defects of Cold Mill Strip Using the Binary Tree Classifier

The defect patterns of cold mill strips can be classified into seven classes: Dull, Oil-drop, Slip, Dent, Scale, Dirt, and Scratch. After preprocessing for the acquired image, we extract six candidate features [8, 10].

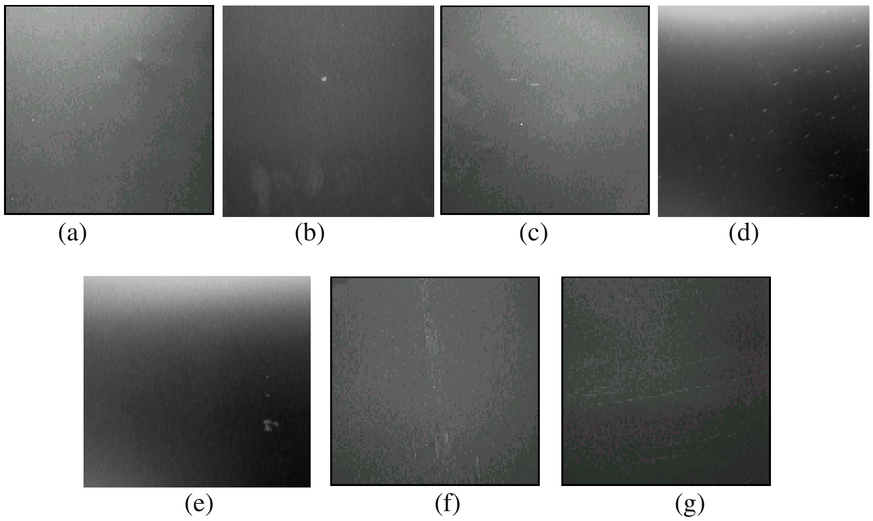
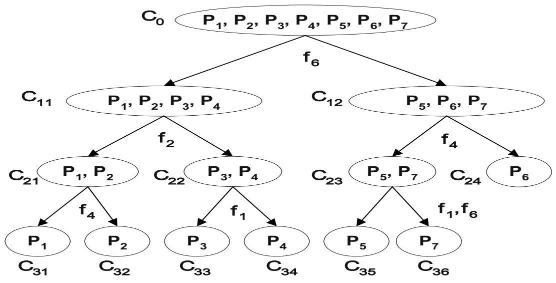


Fig. 2. Defect patterns of cold mill strip (a) dull (b) oil drop (c) slip (d) dent (e) dirt (f) scale (g) scratch

In this paper, geometrical features are selected as candidate features. They are area, area ratio, and compactness. They are not related to the size and direction of the patterns. And, the probabilistic concept of moment has been widely used in pattern recognition as a practical method to extract features of the shape. Among the moment-based features, the information useful for the inspection of the defects of cold mill strips are the length of the longest axis, the ratio of longest axis to shortest axis, and the spread of a pattern [8, 10]. The data used in constructing the binary tree recognizer are the feature vectors extracted from the seven types of standard defect patterns. In constructing the binary tree using GA, the weights in Eq. (4), w_e and w_b, are set to 1. Fig. 3 shows the binary decision tree constructed from standard patterns by GA. Here, P_i is a type of pattern, f_k is a feature, and C_{mn} represents a class at each node.



Symbol	P_1	P_2	P_3	P_4	P_5	P_6	P_7
Pattern	dull	oil	slip	dent	scale	dirt	scratch

Symbol	f_1	f_2	f_3	f_4	f_5	f_6
Feature	def_area	area_ratio	compact-ness	axis_ratio	spread	long_axis

Fig. 3. Binary decision tree constructed with standard patterns

At each node constructed above, the final recognizer is made by training the BP neural network with the selected feature subset. The number of nodes in the input layer is set to the number of the selected features, and the number of nodes in the hidden layer is set to 10. By setting the number of nodes in the output layer to 2, the output layer represents the binary decision.

Table 1 shows the results of recognizing the defect patterns of a cold mill strip using the binary tree recognizer. In this table, the recognition rates of Dent and Slip are very low. However, the linear classification errors are zero at nodes C_0, C_{11}, and C_{22} when constructing the binary decision tree. This means that the standard patterns of Dent and Slip are classified linearly. Because the least number of features that fit to classify standard patterns are selected, if the number of standard patterns is small, the recognizer becomes sensitive to noise.

Table 1. Recognition rate of each defect pattern

Pattern	*No. of Recog. / No. of Patterns*	*Recognition rate (%)*
Dent	0/3	0.0
Dull	6/12	50.0
Oil drop	4/4	100.0
Slip	1/4	25.0
Dirt	2/2	100.0
Scale	19/22	86.4
Scratch	7/8	87.5
Total	39/55	70.9%

4 Conclusions

In this paper, we used a binary decision tree classifier to recognize the defect patterns of a cold mill strip. We have used the cold mill strip of POSCO (Pohang Steel Company), which consists of 55 defect patterns. At each node of the binary tree, GA was used for the selection of the best feature subset, and the linear decision function, also generated by GA, was used to calculate the fitness of a feature subset. There are two advantages of this method. One is that the construction of the binary decision tree and the selection of the best feature subset can be executed automatically for the given patterns. The other is that by designing the fitness function of GA properly, the decision tree can be obtained by considering the balance of the classes as well as the classification error.

Current performance is about 71% of recognition rate. Further studies should be made to design classifiers which have more generalization capabilities and feature extraction methods which are mutual helpful for the recognition of the defect pattern of a cold mill strip.

References

1. Jung S.W., Park G.T.: Design and Application of Binary Decision Tree using Genetic Algorithm. Journal of the KITE, 33(6) (1996) 1122-1130
2. Brill F., Brown D., Martin W.: Fast Genetic Selection of Features for Neural Network Classifiers. IEEE Trans. Neural Networks, 32 (1992) 324-328
3. Yao L.: Nonparametric Learning of Decision Regions via the Genetic Algorithm. IEEE Trans. Sys. Man Cybern., 26 (1996) 313-321
4. Mui J.K., Fu K.S.: Automated Classification of Nucleated Blood Cells using a Binary Tree Classifier. IEEE Trans. Pattern Analysis Machine Intelligence, 5 (1980) 429-443
5. Safavian S.R., Landgrebe D.: A Survey of Decision Tree Classifier Methodology. IEEE Trans. Sys. Man Cybern., 21 (1991) 660-674
6. Payne H.J., Meisel W.S.: An Algorithm for Constructing Optimal Binary Decision Trees. IEEE Trans. Computers, 26 (1997) 905-916
7. Swain P.H., Hauska H.: The Decision Tree Classifier: Design and Potential. IEEE Trans. Geosci. Elec., 15 (1997) 142-147
8. Kim K.M.: Development of Surface Defect Inspection Algorithms for Cold Mill Strip. Journal of Control, Automation and Systems Eng., 3 (1997) 179-186
9. Babu G.P., Murty M.N.: Clustering with Evolution Strategies. Pattern Recognition, 27 (1994) 321-329
10. Kim K.M.: Design of Hierarchical Classifier for Classifying Defects of Cold Mill Strip using Neural Network. Journal of Control, Automation and Systems Eng., 4 (1998) 499-505

Genetic Algorithm-Induced Optimal Blackjack Strategies in Noisy Settings

Ron Coleman[1] and Matthew A. Johnson[2]

[1] Computer Science Department
Marist College
Poughkeepsie, New York, United States
ron.coleman@marist.edu

[2] Informeta, LLC
Poughkeepsie, New York, United States
matthew.johnson@informeta.net

Abstract. This paper investigates a new class of parameterized hybrid genetic algorithms called LV(k) that learns to play Blackjack not only successfully and in some cases, unconventionally, compared to the professional Basic Strategy. For the most promising k class, namely, k=1, we show that 19 instances of these new strategies consistently exceeds previous A.I. efforts by as much as sixteen-fold in game returns. Furthermore, LV(1) is more robust than the Basic Strategy under additive spectral noise by about 1 db at the "flutter" boundary of about –16 db noise intensity where all strategies appear to exhibit statistically significant distortion.

1 Introduction

Blackjack or "21" is a simple card game [Wong 1992] that is very popular with the public, casinos, mathematicians, game theorists, and computer scientists. Blackjack is also one of the few games in which the House does not, in theory, have an overwhelming advantage. For instance, provided one plays "correctly" according to the optimal Basic Strategy (B/S) to maximize returns, a player using no other method (e.g., no card counting) can expect -0.49% returns in games with four decks, no double after splits, and the dealer standing on soft 17. [Griffin 1999] For games with an infinite deck, a player can expect -0.65% returns using the B/S. [Griffin 1999] However, a B/S (like the one in the Appendix) assumes perfect play. Yet longstanding anecdotal evidence suggests people do not follow the rules systematically, if at all, even when they know they should. [Thorp 1966] Thus, inconsistent or "noisy" application of the B/S limits player opportunities in practice.

In this paper, we investigate the role of the noise for Blackjack through a new class of parameterized hybrid genetic algorithms [Goldberg 1989], LV(k), that learn to play Blackjack competitively and in some cases unconventionally, exceeding previous A.I. efforts by as much as sixteen-fold in game returns. The k parameter in LV(k) is the number of simultaneous hands the heuristic optimizes conditioned on previously learned k-type hands. We show furthermore these LV(k)-induced strategies are statistically more stable by about 1 db when subjected to additive white or uniform noise which has constant power spectral density and to additive pink or 1/f noise which has power spectral density that decreases with increasing frequency. [Schroeder 1991] The motivation for using white and 1/f noise was primarily to contrast the strategy responses in the –40 db to –1 db range using the noiseless game returns as experimental controls for the B/S versus several instances of the most promising LV(k) class, namely, k=1. There is also evidence that 1/f fluctuations, which have been identified many different chaotic phenomena of nature, also play a role in human cognition and psychiatry [Pressing 1999], although the implications for Blackjack are an open research issue.

2 Related Work

Griffin (1999) gives a game theoretic analysis of noiseless play opportunities in Blackjack. Griffin includes various mathematical approximations and simulation results as well as references to scholarly studies, including those of Baldwin (1956) and Braun (1977) who first developed the Optimal Basic Strategy. Griffin does not, however, contemplate machine learning nor additive noise applications. It seems the earliest references in the A.I. literature to machine-learning techniques for Blackjack appear to be genetic in nature due to Gryk (1993) and Yun (1997). However, according to Caverlee (2000) these early efforts were "hampered both by computing limitations and inadequate problem specification." Thus, Caverlee developed a clearer problem specification centered on the idea that a gene could represent a single player decision, d={stay, hit, double, split}, conditioned on the player's hand and the dealer's upcard. There are $h \times w$ such genes forming a genome (or chromosome) of h player hands by w dealer upcards representing a global strategy of all possible player decision configurations. The approach then uses a standard (i.e., non-hybrid) genetic algorithm to search the global space of Blackjack strategies, optimizing ~300 player decisions simultaneously for maximum mean return. While global search is relatively straightforward to set up, it suffers from combinatorial explosion as the search space consists of approximately 10^{220} possibilities. Indeed, even Caverlee faced computational barriers and in any event, did not find solutions whose mean returns were competitive with B/S. These efforts didn't consider stability in noisy settings.

3 Methods

3.1 Blackjack and the Game Simulator

The game simulator plays N games assuming an infinite deck. The dealer stands on soft 17 or higher and no doubles after splits are allowed. Surrender and insurance are not allowed. Blackjack pays 1.5:1 and all other bets pay 1:1. The simulator uses a programmable strategy table of 420 decisions: h=42 player hands by w=10 dealer upcards. However, since 21 must stay, only 370 rules can be programmed. The B/S in the Appendix has the same format except by convention it is condensed, e.g., “17+” is “17”, “18”, “19”, “20”, and “21”.

3.2 LV(k) Heuristic

LV(k) is a hybrid GA that uses an iterative search method. It resembles hill climbing [Russell 2000] where k is the number of simultaneous player hands the heuristic optimizes using a GA in the innermost (seed) loop. (See pseudo code below.) The middle (j) loop controls which k-type player hands the GA visits and it also selects the best GA solution using a majority-voting scheme. (Each GA solution from the innermost loop “votes” on k player decisions and in the event of ties, middle loop chooses the solution with the best total return from 20,000 out-of-sample games. Finally, the outmost (i) loop controls the dealer upcard whereby the card in the “hole”, that is, the hidden card, varies randomly as cards drawn from the simulator’s deck. The middle and outermost loops of LV(k) represent two inherent constraints of player decisions in Blackjack which are independent of k and which have not been exploited in earlier studies.

For the first constraint, which involves elimination of impossible next-card scenarios, suppose we label the possible player hands “17+” through” 2,2” as 1...h. (For the strategies in the Appendix, h=26.) Suppose at the start of the game, a player has two cards, the label of which is n such that $1 \leq n \leq h$. The first constraint stipulates that the next card in the deck can only result in an m hand, n>m. In other words, player hands move down in label number, never up, on the next card. For instance, a player holding a “15” hand (labeled 3) may get a “16” or higher (labeled 2 or 1) on the next card. However getting a “14” (labeled 4) on the next card is logically impossible.

For the second constraint observe that the play decision is conditioned only on the player’s hand and dealer’s upcard, both of which are fixed and known. The dealer’s card in the “hole” and the next card in the deck are uncertain. The second constraint stipulates that only decisions for a given upcard, i.e., in a given column in the B/S table, are relevant for a player decision. For instance, a player holding a 15 against a dealer’s 3 is concerned about whether to hit, stay, or double down in that situation only. Any other dealer upcard is not relevant.

Using these two constraints, LV(k) learns from scratch the rule for hand labeled n=1 (i.e., hand=”17+”) and a given dealer upcard. Having selected a decision for n=1 and a given dealer upcard, LV(k) learns the rule for n=2 (i.e., hand=”16”) given the same dealer upcard plus the rule LV(k) previously learned for n=1. LV(k) proceeds in this manner until the entire dealer upcard strategy has been learned for n={1...h}. LV(k) repeats the process for the next dealer upcard. This is the procedure for k=1 hand. The dealer upcards are independent as we indicated to earlier and we can apply the same algorithm for k>1 to optimize more than one hand simultaneously. For example, in the first iteration the middle j loop optimizes hands “17+” and “16” simultaneously for an upcard, U_i. In general, we select k such that h *mod* k = 0. For lower order k, the search space is reduced to $O(2^{2k}/k)$ while the run-time complexity is

O(h/k) which can be analyzed much faster than $O(2^{2\times370})$ and O(1) respectively for the global search method. The triple-nested heuristic in pseudo code is below:

```
For i=1 to w of dealer upcards, $U_i$
  For j = 1 to h step k
    For seed=1..5
         GA-optimize (seed, $[n_j \ldots n_{j+k-1}]^T$, $U_i$, 1000*k)
         Test GA solution on 20,000 games
    End-for
    Select most frequent strategy by
      majority vote or by best total return if tied
  End-for
End-for
```

3.3 GA Parameters

The objective function optimizes total return from1,000×k games to ensure reliable statistics. By "total return" we mean the sum of profits and losses from all the hands of all the games. We use a mutation rate of 10%, crossover rate of 90%, and a "simple" (as opposed to steady-state) GA, which builds the gene pool anew with each generation. The GA stops after 100 generations or when the diversity reaches zero. The genome schema consists of the decisions, $[n_j \ldots n_{j+k-1}]$ for k hands and there are T=10 genomes in the gene pool, although the playable hands grow by k with each iteration of the middle (j) loop. We run the above process 19 times using a different random seed, which in turn produces 19 different LV(1) strategies. We label each strategy A*xx* where xx is the seed number.

The LV(k) heuristic induces Blackjack strategies that are statically or lexically distinct from the B/S. We measure the discrepancy between the B/S and the LV(k) strategies using the *error-correcting Levenshtein distance* [Fu 1982]. In other words, we can think of a strategy as a string with length 420. Thus the Levenshtein distance, ∇_k, is the number of corrections in the form of direct player decision substitutions to make LV(k) identical to the B/S. Our interest in this metric is to determine if there's a relationship between k=1 type strategies and the strategies mean game returns.

3.4 Spectral Noise

In a noiseless setting, the player executes the strategy's implied decision, d, with complete certainty, dependent only on the player's hand and the dealer upcard. In a noisy setting, the execution is further dependent on a noise fluctuation model such that implied decision is now a random variable in the following manner. Let A_d represent the probability of accepting the implied decision and G_d, the probability of generating the implied decision given d is initially rejected. That is, the probability of deciding d is $A_d + G_d(1 - A_d)$ such that $0 < A_d \leq 1$ and $0<G_d<1$. We thus have,

$$A_d + (1 - A_d) \Sigma_{x\in X} G_x = 1 \tag{1}$$

for all possible legal actions, X, which admits the possibility of regenerating the implied decision, d. Simplifying this further, we have $A_d + I_d = 1$ where $I_d=(1–A_d)\ \Sigma_x\ G_x$ is the linearly "additive" noise intensity. In the noiseless scenario, $A_d=1$ and $I_d=0$. However, in the noisy case, $0<A_d<1$ means there is a finite probability, $A_x = I_d - (1-A_d)\ G_d$ of deciding some other legal decision, x. If a white noise process generates I_d, then the power, P_d, is uniform across all decision spectra, namely,

$$P_d = r^2 \qquad (2)$$

In other words, r is the "base noise" intensity, which is constant for all decisions. If a 1/f noise process generates I_d, we have $I_d \sim r / f_d$ where f_d is the spectral frequency of d in the strategy and f_0 is the first harmonic or lowest frequency decision. In this case, the power is

$$P_d = (r\ f_0 / f_d)^2 \qquad (3)$$

Following convention, we express the noise amplitude in decibels, $10 \times \log I_d / I_0$ where I_0 is the arbitrary reference intensity which for convenience sake we set to 1.

3.5 Nonparametric Tests

To conservatively estimate p values, we use two standard, nonparametric hypothesis tests: resampling and the binomial test.

Resampling Tests. We follow Simon (1995) and apply a resampling algorithm for measured data to estimate p values in two ways for: (1) the difference in expected return between the B/S strategy and the LV(1) strategy; (2) the difference in expected return between the noiseless and noisy setting. In the first case, we have H_0:E[B/S]=E[LV(k)]=0 and the alternative, H_1:E[B/S]≠E[LV(1)]. In the second case we have H_0:E[S]=E[S+I_d] and the alternative, H_1:E[S]≠E[S+I_d]. S is the strategy, B/S or LV(1). In each of these two cases, we resample 100,000 earnings N=1,000 times to find the fraction of randomly sampled earnings that exceed the measured earning to estimate the p value with accuracy of 10^{-3}.

Binomial Tests. We apply the binomial test to estimate p values in the difference in noise intensity between the B/S and LV(1) strategies where each exhibits *flutter*, i.e., statistically significant distortion in mean return. We test for flutter as follows. Let $I_{B/S}$ and $I_{LV(1)}$ be the noise intensity where flutter occurs for B/S and LV(1) on a Bernoulli trial of 100,000 games. The trial is successful and LV(k) more robust than B/S if $I_{LV(1)} > I_{B/S}$. The trial is unsuccessful otherwise. In N trials, the number of successes is s and the number of failures is s*. The null hypothesis is H_0:s≤s* and the alternative, H_1:s>s*. We use the test statistic, T_1, which follows a binomial distribution:

$$T_1 = (s - ½\ N) / \sqrt{(¼\ N)} \qquad (4)$$

4 Results

The experimental data that follows in this section is based on running approximately 45,000 sets of 100,000 out-of-sample games or about 4.5 billion games total on the B/S and the learned LV(1) strategies. The Appendix gives two examples of the LV(1) strategies: A26, the "worst" and A91, the "best" in a noiseless setting. The noiseless mean return for the professional B/S (measured in the game simulator) is 0.60% with a standard error of 0.36%. The mean returns for all 19 instances of the LV(1) strategies range from –1.05% to –0.48%, all with standard errors of 0.35%. The p values range from 0.242 to 0.887. In other words, the LV(1) strategies are statistically indistinguishable from the B/S. Caverlee (2000) which in our opinion best represents the global search approach gives a mean return of –7.43% and a $p<10^{-12}$ which we estimate "roughly" using its standard error of 0.12%. Thus, global search is not competitive with B/S and moreover, underperforms LV(1) by 7X – 16X in game returns.

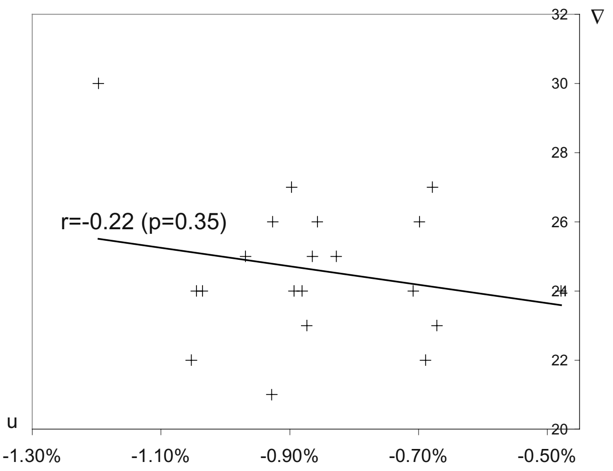


Fig. 1. LV(1) v. ∇_k scatter plot on 100,000 games in noiseless setting.

Table 1 contains a sampling of the stability results for the LV(1) strategies in Figure 1. The columns give the mean noise intensity in db where the difference in game returns between the strategy in a noiseless and noisy setting become statistically significant. The "Dif" column is the noise intensity difference.

5 Conclusions

The two sample LV(k)-induced strategies in the Appendix are indicative of k=1 learning of all 19 strategies. Inspection suggests that the primary difference between the B/S and LV(1) strategy is the generally higher frequency of double downs in LV(1). This is suggestive of a risk-prone style of play, which is unconventional, at variance with Blackjack game theory analysis and counterintuitive for robustness. Future research proposes to study this connection between aggressiveness and stability for a greater variety of noise fluctuation models.

Table 1. Sampling of strategies responses under white and 1/f noise.

	White noise						1/f noise					
Strat	s	trials	p	$\bar{I}_{B/S}$	$\bar{I}_{LV(1)}$	Dif	s	trials	p	$\bar{I}_{B/S}$	$\bar{I}_{LV(1)}$	Dif
A06	220	400	0.020	-16.86	-16.00	0.86	174	300	0.002	-16.26	-15.19	1.07
A11	143	250	0.001	-16.95	-16.13	0.82	145	200	$<10^{-10}$	-16.29	-15.17	1.12
A16	142	250	0.013	-16.95	-16.07	0.88	137	200	$<10^{-7}$	-16.29	-15.05	1.24
A21	141	250	0.018	-16.95	-16.15	0.80	129	200	$<10^{-4}$	-16.29	-15.23	1.06
A26	147	250	0.002	-16.95	-16.09	0.86	136	200	$<10^{-7}$	-16.29	-14.99	1.30

References

1. Baldwin, R., et al, 1956. The Optimum Strategy in Blackjack, *J. Amer Stat. Assoc.*
2. Braun, J., c. 1977. The Development and Analysis of Winning Strategies for Casino Blackjack, IBM private report.
3. Caverlee, J.B., 2000. A Genetic Algorithm Approach to Discovering an Optimal Blackjack Strategy, *Genetic Algorithms and Genetic Programming at Standford 2000.* J. Koza ed.
4. Fu, K.S., 1982. *Syntactic Pattern Recognition and Applications*, Prentice Hall.
5. Goldberg, D.E., 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley.
6. Griffin, P., 1999. *The Theory of Blackjack.* Huntington
7. Pressing, J. 1999. Sources for 1/f noise effects in human cognition and performance. *Proc. 4th Conf. of the Australasian Cognitive Science Society*, Newcastle
8. Schroeder, M., 1991. *Fractals, Chaos, Power Laws.* Freeman.
9. Simon, J.L., 1995. Resampling: The New Statistics, Resampling Stats Inc.
10. Russell, S., Norvig, P., 2003. *Artificial Intelligence*, Prentice Hall, 2nd ed
11. Thorp. E.O., 1966. *Beat the Dealer.*
12. Wong, S. 1992. *Basic Blackjack.* Pi Yee.
13. Yun, Y., 1997. Finding an Optimal Blackjack Strategy using Genetic Algorithms, *Genetic Algorithms and Genetic Programming at Standford 1997.* J. Koza ed.

Appendix: Professional B/S v. Sample LV(1) Strategies

The B/S gives the player decision as “X” and LV(1) as “Y” where there are discrepancies in the shaded “X/Y” decisions. The arrow gives the direction of LV(k) learning.

A26: $\nabla_1 = 30$

	Your hand	Dealer’s upcard									
		2	3	4	5	6	7	8	9	10	A
1	17+	S	S	S	S	S	S	S	S	S	S
2	16	S	S	S	S	S	H	H	H	H/S	H
3	15	S	S	S	S	S	H	H	H	H	H
4	14	S	S	S	S	S	H	H	H	H	H
5	13	S	S	S	S	S	H	H	H	H	H
6	12	H	H	S/H	S	S	H	H	H	H	H
7	11	D	D	D	D	D	D	D	D	D	H/D
8	10	D	D	D	D	D	D	D	D	H/D	H
9	9	H/D	D	D	D	D	H	H	H	H	H
10	5-8	H	H	H	H	H/D	H	H	H	H	H
11	A 8-10	S	S	S	S/D	S/D	S	S	S	S	S
12	A, 7	S/D	D	D	D	D	S	S	H	H	H/S
13	A, 6	H/D	D	D	D	D	H/D	H	H	H	H
14	A, 5	H/D	H/D	D	D	D	H	H	H	H	H
15	A, 4	H/D	H/D	D	D	D	H	H	H	H	H
16	A, 3	H/D	H/D	H/D	D	D	H	H	H	H	H
17	A, 2	H/D	H/D	H/D	D	D	H	H	H	H	H
18	A,A,8,8	P	P	P	P	P	P	P	P	P	P
19	10,10	S	S	S	S	S	S	S	S	S	S
20	9, 9	P	P	P	P	P	S/P	P	P	S	S
21	7, 7	P	P	P	P	P	H/P	H	H	H	H
22	6, 6	P	P	P	P	P	H	H	H	H	H
23	5, 5	D	D	D	D	D	D	D	D	H/D	H
24	4, 4	H	H	H	P/H	P/D	H	H	H	H	H
25	3, 3	P	P/H	P	P	P	P	H/P	H	H	H
26	2, 2	P	P/H	P	P	P	P	H	H	H	H

A91: $\nabla_1 = 24$

	Your hand	Dealer’s upcard									
		2	3	4	5	6	7	8	9	10	A
1	17+	S	S	S	S	S	S	S	S	S	S
2	16	S	S	S	S	S	H	H	H	H	H
3	15	S	S	S	S	S	H	H	H	H	H
4	14	S	S	S	S	S	H	H	H	H	H
5	13	S	S	S	S	S	H	H	H	H	H
6	12	H	H/S	S/H	S	S/H	H	H	H	H	H
7	11	D	D	D	D	D	D	D	D	D	H/D
8	10	D	D	D	D	D	D	D	D	H/D	H
9	9	H/D	D	D	D	D	H	H	H	H	H
10	5-8	H	H	H	H	H/D	H	H	H	H	H
11	A 8-10	S	S	S	S/D	S/D	S	S	S	S	S
12	A, 7	S/D	D	D	D	D	S	S	H	H	H
13	A, 6	H/D	D	D	D	D	H	H	H	H	H
14	A, 5	H	H/D	D	D	D	H	H	H	H	H
15	A, 4	H	H/D	D	D	D	H	H	H	H	H
16	A, 3	H/D	H/D	H/D	D	D	H	H	H	H	H
17	A, 2	H	H/D	H/D	D	D	H	H	H	H	H
18	A,A,8,8	P	P	P	P	P	P	P	P	P	P
19	10,10	S	S	S	S	S	S	S	S	S	S
20	9, 9	P	P	P	P	P	S	P	P	S	S
21	7, 7	P	P	P	P	P	P/H	H	H	H	H
22	6, 6	P	P	P	P	P/H	H	H	H	H	H
23	5, 5	D	D	D	D	D	D	D	D	H/D	H
24	4, 4	H	H	H	P/H	P/D	H	H	H	H	H
25	3, 3	P/H	P	P	P	P	P	H	H	H	H
26	2, 2	P	P	P	P	P	P	H	H	H	H

Robust Semantic for an Evolved Genetic Algorithm-Based Machine Learning

Yamina Mohamed Ben Ali [1] and M. Tayeb Laskri [2]

[1] Laboratoire de Recherche en Intelligence Artificielle, Institut Informatique, Universite Badji Mokhtar BP 12, Annaba, Algérie
Benaliyam2@yahoo.fr

[2] Laboratoire de Recherche en Intelligence Artificielle, Institut Informatique, Universite Badji Mokhtar BP 12, Annaba, Algérie
laskri@yahoo.com

Abstract. The aim of this paper is to introduce a robust semantic to genetic learning behavior. The conceptualization of new genetic machine learning relies on the specification model of Entity-Relation. According to this perspective, the learning behavior is decomposed in two evolution dimensions. In addition to some added concepts, the model uses the learning metaphor of coarse adaptability. The configuration of the system leads to a decentralized architecture where entities and interactions provide more reality to the overall genetic machinery. Based upon an adaptation function, the learning put emphasizes on adjustments of mutation rates through generations. This landscape draws the best strategy to control the intensity of convergence velocity along of evolution.

1 Introduction

Efficient learning in Evolutionary Algorithms [2,3,16] is implicitly described by the adaptation behavior regarded as simple or complex feature [7]. Its role is to ensure more elasticity and flexibility to potential parameters influencing closely convergence performance. Actually, the adaptation is most present in all EA theories like: Evolutionary Strategies (ES) [4,5,6] and Evolutionary Programming [8,11].

Although Genetic Algorithms (GA) are nowadays more adaptive, their applicability and suitability still relative to the considered class of problems. However, the adaptation operates on different levels of a GA (Population-Individual-Component). In this sense, there exists many adaptive GA [1,9,10,12,13,14,15].
The purpose of this article is to improve both genetic reasoning and convergence performance, by hybridizing conceptually all genetic learning levels. In other words, we aim to extend the field of genetic applications by proposing a robust evolution semantic. Then, as a genetic machine learning, our model encloses a high learning layer and overcome traditional shortcomings.

2 Genetic Machine Learning Specifications

A machine learning based upon genetic reasoning introduces a new resolution trend in genetic community. It represents the semantic layer needed to give more applicability and generality to genetic algorithms. Thus, by means of adaptability behavior, genetic algorithm as evolutionary machine learning could be able to model large range of real world problems.

However, to better understand the global mechanism of this machinery, we must consider the behind theory of ***adaptability***. In the present context, given that the evolution system is basically described in terms of entities and interactions, the adaptability complexity will then draw the coarse side of evolution (see Fig.1).

2.1 Basic Concepts and Terminology

Definition 1 *[Adaptability]:* adaptability coarsely defined is an adjustments methodology well described by an entities space at 2-dimensions.

To explain our evolution approach adaptability-based, some genetic requirements must be fulfilled. The parental population conceptually describes by entities constitutes the basic variable evolutionary dimension. The learning behavior semantically defined by interactions dimension, constitutes the enhancement of the system and yields to an intelligent behavior.

Practically, coarse adaptability attempts to benefit from the main characteristic of parents population. In fact, based on parents' fitness and structures as genetic resources, the evolutionary machine, via adaptability, attempts to guide all evolution steps. To accurately enable convergence performance, the adaptability put emphasis on the specification model of *Entity-Relation*. Then, to better formalize the previous metaphor, some concepts should be defined.

Definition 2 *[entity]*: entity is a conceptual description of an individual chromosome. Its role depends strongly on the interfered dimension (level).

Definition 3 *[interaction]*: interaction is a virtual relation used to denote a semantic link between either components of an entity or two entities.

In what follows, a detailed description of the evolution behavior is given under two main sections. Each section explains different adjustments occurred on an adaptive dimension. It is also interesting to say that a dimension is either:

- An abstract view (macro dimension), where all entities are solely evaluated in function of their fitness values, and where no interactions exist.
- A component view (micro dimension), where all components entities establish logical and semantic links over components. These cooperative interactions will determine the suited mutation rates.

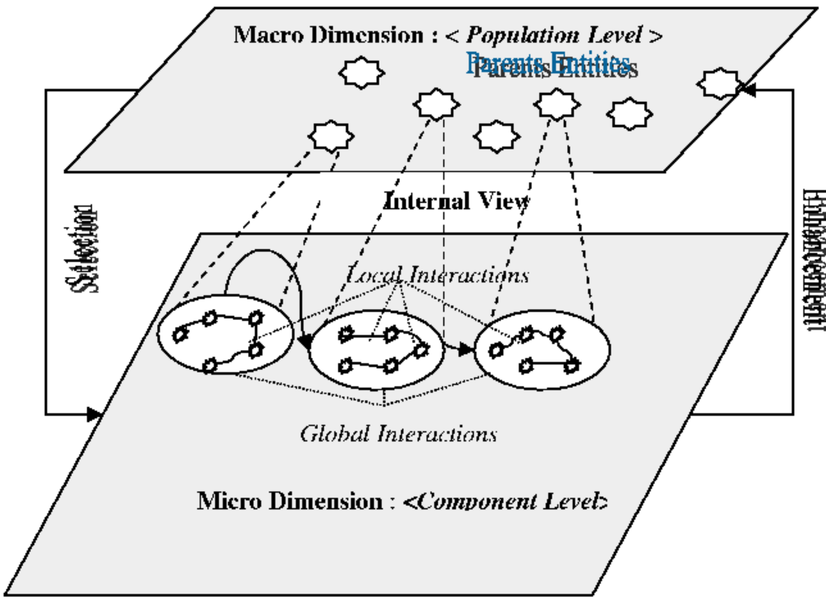


Fig. 1. Conceptual View of Genetic Learning Machine

Since an overall focus is on entities, the goal of coarse adaptability is to distribute the learning task according to 2-dimensions evolution:

- First dimension: all entities evolve independently from each other in order to quantify the pertinent entities able to survive.
- Second dimension: all pertinent entities learn at fine granularity (component). According to their micro description, components of an entity cooperate by local interactions to produce a best offspring entity.

The evolution is then a progressive learning aiming to produce, at each generation, best parents' entities able to self-learn in both a cooperative and competitive way.

2.2 Evolution Macro Dimension

The macro evolution points out the entities population. At this evolution level, all parents' entities are evaluated quantitatively in order to *select pertinent* ones. Like such strategy is important to both build the best entities pool and determine the rate selection required at the next dimension.

Therefore, according to a ranking scheme, the entities population $\Omega(t) = \{E_1, E_2, E_3, \ldots, E_\mu\}$ is sorted by ascendant order. To each entity $(E_i, f_i) \rightarrow \varpi(E_i)$, where f_i is the fitness value (objective value), is assigned a *life period* ϖ. Indeed, this strategy parameter represents the difference between the entity fitness value and the fitness of the best entity (better one). This design is usually useful to achieve the selection criterion and even the probability that enables entity to continue to survive or dead.

Thus, compared to the high life period $\overline{\varpi(E)}$ (relative to the oldest entity that has worst fitness), it is now possible to assign a survival probability to each entity like:

$$\rho\ (E)\ = \left(1-\gamma * \frac{\varpi(E)}{\overline{\varpi(E)}}\right)^2 \tag{1}$$

So, to access to the second evolution dimension and survive, an entity must satisfy the selection criterion rule: $\rho(E) \geq 0.5$. This is paramount to perform the rate selection β performed as the sum of parents satisfying the selection criterion.

Thus, by β means as population qualitative criterion, all entities with adequate selection probabilities could participate to mating pool.

2.3 Evolution Micro Dimension

The micro evolution supposes that only a category of best entities is considered. At this level, an entity is a set of independent components semantically linked. Through the specific relation of *interaction,* the components of an entity establish local interactions. Thus, they cooperate by communicating their adaptation velocities called learning rates. This sort of *heritage mechanism* enables also global interactions between different entities, given that the last performed learning rate of an entity E_i is itself the learning rate of the E_{i+1} first component.

We state that the learning task is applied synchronously to all pairs of mating entities. As consequence, the *enhancement* of the offspring population is subject to the variability of all applied learning rates which directly influence the components mutations rates. The global parameter accountable of all these modifications is designed by the *adaptation function* which outperforms the population quality.

$$\theta(t)=\alpha\,(t)\ {}^{1}\!/\!\sqrt{\beta(\mathrm{t})} \tag{2}$$

$\alpha(t)$ is the expected neighboring distance between best entities.

The heritage behavior is always exhibits provided that the maximal learning threshold was not achieved. When this situation happens, it would become necessary to inhibit the interaction behavior between all entities. In this case, entities should evolve without any communication, and then the role of learning rate is useless. This is meaningful in order to decrease the diversity rate caused by learning rates.

3 Conclusion

The presented evolutionary approach proposed in this article parameterizes the genetic evolution by improving the learning behavior. This intends to embody a semantic layer of machine learning guided by coarse adaptability. Based on macro and micro dimensions, the adaptability implements a new view of entity-relation to conceptually model the genetic machine learning. Several concepts have been also introduced to describe the above architecture. The hide side of this semantic is the specification of a new convergence trend which based on adaptation function enables accurate performance. However, this landscape will be eventually model by the fine adaptability as perspective to give more convergence velocity and scalability to genetic algorithms.

References

1. Bäck, T.: An overview of parameter control methods by self-adaptation in evolutionary algorithms. Foundamenta Informaticae, Vol. 35 (1-4), (1998) 51-66
2. Bäck, T., Fogel, D.B., Michalewicz, Z.: Evolutionary computation 2: advanced algorithms and operators. Institute of physics publishing (2000)
3. Fogel, D. B.: Evolutionary computing. IEEE Press (2002)
4. Beyer, H.-G.: Towards a theory of Evolution Strategies: Self-adaptation. Evolutionary Computation, vol. 3(3) (1996) 311-347
5. Beyer, H.-G., Deb, K.: On self-adaptive features in real-parameter evolutionary algorithms. IEEE Transaction on Evolutionary Computation Vol. 5 (3) (2001) 250-270
6. Beyer, H.-G.: The theory of evolution strategies. Natural Computing series, Springer, Heidelberg (2001)
7. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE transaction on Evolutionary computation Vol. 3 (1999) 124-141
8. Fogel, L.G., Angeline, P.J., Fogel, D.B.: An evolutionary programming approach to self-adaptation on finite state machines. In J.R Mac Donnell, R.G Reynolds and D.B Fogel (Eds.). Proceeding on the fourth international conference on evolutionary programming (1995) 355-365
9. Garnier, J., Kallel, L., Schoenauer, M.: Rigorous hitting times for binary mutations. Evolutionary Computation 7 (1999) 173-203
10. Hinterding, R., Michalewicz, Z., Peachey, T.: Self adaptive genetic algorithm for numeric functions. Proceedings of the Fourth Conference on Parallel Problem Solving from Nature, Springer Verlag (1996) 420– 429
11. Liang, K.-H., Yao, X., Liu, Y., Newton, C.: Dynamic control of adaptive parameters in Evolutionary Programming. Second Asia Pacific Conference on simulated evolution and learning, Lecture Notes in Artificial Intelligence, Vol. 1585, Springer-Verlag, Berlin (1998) 42-49
12. Spears, W.M.: Adapting crossover in evolutionary algorithms. In Mc Donnel, J., Reynolds R., and Fogel, D.(eds), Proceedings of the Fourth Annual Conference on Evolutionary Programming, MIT Press (1995) 367–384
13. Spears, W.M.: Evolutionary algorithms: The role of Mutation and Recombination. Springer, Heidelberg (2000)
14. Smith, R.E., Smuda, E.: Adaptively resizing populations: Algorithm, analysis and first results. Complex Systems Vol. 9(1), (1995) 47–72
15. Thierens, D.: Adaptive mutation rate control schemes in genetic algorithms. http://citeseer.nj.nec.com/thierens02adaptive.html, (2002).
16. Wegener, I.: Theoretical aspects of evolutionary algorithms. Twenty-eight International colloquium on automata, language, and programming (ICALP) (2001)

A Hybrid Neural-Markov Approach for Learning to Compose Music by Example

Karsten Verbeurgt, Mikhail Fayer, and Michael Dinolfo

Department of Computer Science
State University of New York at New Paltz
New Paltz, NY, 12561
`Verbeurg@cs.newpaltz.edu`

Abstract. In this paper we introduce a hybrid approach to autonomous music composition by example. Our approach utilizes pattern recognition, Markov chains, and neural networks. We first extract patterns from existing musical training sequences, and then construct a Markov chain based on these patterns with each state corresponding to a pattern. We then use a neural network to learn which shifts of pitch and duration are allowed for each pattern in the training sequences. Using this hybrid model, we compose novel musical sequences.

Keywords: Neural Nets; Applications of AI in Information Processing; Machine Learning

1 Introduction

Since the early days of digital computers, researchers have aspired to write computer programs that compose music. This is in part due to the fact that music is an art form that is amenable to formal representation in a score that can be encoded discretely in a computer. An additional motivation is found in the fact that many composers have used algorithmic techniques as a compositional tool, giving hope that a composition process can be captured in an algorithm. Nevertheless, music composition remains a difficult challenge even for the most skilled human practitioners. It would appear, then, that some aspects of composition can be captured algorithmically, but perhaps that a higher level of creativity is beyond the realm of computation. In this work, we propose a hybrid approach that uses musical training sequences to learn to compose, thus incorporating human-composed musical patterns in an algorithmic composition technique.

Previous work on algorithmic music composition has explored various techniques, including neural networks, evolutionary computation, formal grammars, automata, Markov chains, and fractals [1,2]. For a more extensive discussion of previous work, see [4]. In most prior work, the programmer encodes basic music theory rules that are used to guide composition. In contrast, in this work we do not provide any music theory rules to the program, but instead *learn* them from the training sequences. Our hope is that this approach can not only capture

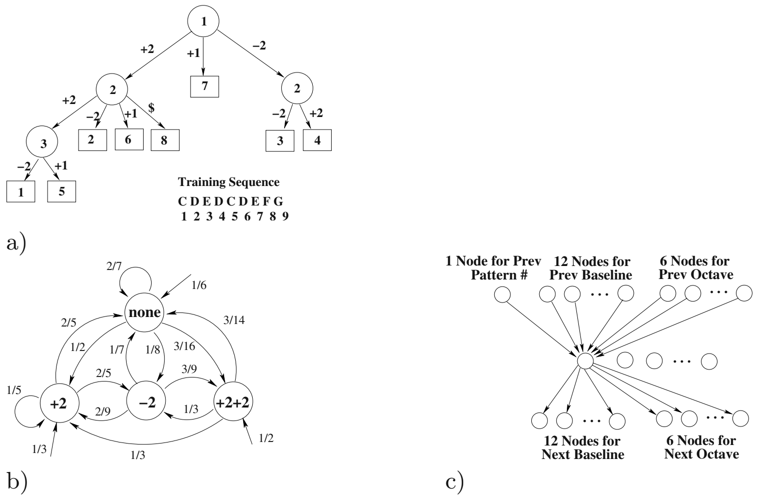


Fig. 1. a) The edges of the suffix tree represent the intervals between successive notes. Leaves correspond to unique occurrences of a pattern, and internal nodes correspond to patterns that repeat. b) A Markov chain in which nodes represent patterns of intervals between successive notes, and transitions represent the probability of patterns occurring in sequence. c) A neural network that takes as input the previous state of the Markov model, along with its baselines for pitch and duration, and generates a baseline pitch and duration for the current state.

music theory rules, but more importantly the exceptions to the rules that create variety in a composition.

In our approach for learning to compose music by example, we first extract patterns, or *motives*, from the training sequences. We consider a pattern to be a sequence of intervals that may start at different *baseline pitches*, and have different *baseline durations*. We then construct a Markov chain in which each state represents a pattern, and transitions between states represent allowable sequencing of patterns. We train a neural network to select the baseline pitch and duration for the pattern of intervals represented by each state of the Markov chain. We use this model to compose novel musical sequences.

2 Suffix Trees for Pattern Extraction

The first step in our method of learning to compose music from existing pieces is to extract patterns from the pieces. We use a *suffix tree* data structure for the pattern extraction. For a detailed description, see [4]. The edges in the tree in Fig. 1 a) represent the intervals, in half-steps, between successive notes. Each node in the tree thus corresponds to a pattern of interval steps. Note that in the suffix tree, all internal nodes correspond to patterns that repeat at least once in the training sequences, and leaves correspond to exactly one occurrence of the pattern. The beginning position of the pattern in the training sequence is stored at each leaf, and the baseline pitch for the sequence is given by the note at that position.

3 Composition Using a Markov Chain

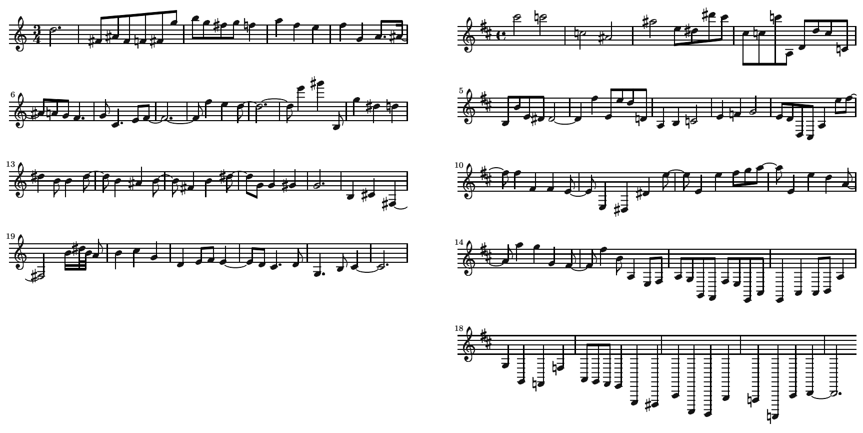
Once the patterns in a set of training pieces have been extracted, we use these patterns as the states in a Markov chain (see Fig. 1 b)). The state labeled "none" in the diagram corresponds a single note, so there is no interval for successive notes in the pattern. The transitions model allowable sequencing of patterns in the training pieces, and transition probabilities are set to reflect the frequencies with which each pattern follows each other in the training sequences. The initial states of the model are the nodes whose corresponding pattern begins one or more of the training sequences. The baseline note of the pattern may, however, differ from the actual baselines of the training pieces (see section 4 for a discussion on generating baselines.) We do not specify final states for the model, but instead terminate the generated sequences once they reach 75% of the average length of the training sequences, and arrive a whole, half or quarter note on the root or fifth of the key. The justification for this is that most pieces end on either the root or fifth of the key on a note that is not transitory.

In previous work that uses Markov chains for music composition, there is one state for each note [1,2], and the transition probabilities determine the probability that a note of one value follows a note of another value. In contrast, in this work, each state represents potentially complex patterns of intervals between successive notes, allowing for a richer structure.

Our Markov model could in fact be viewed as a higher-order Markov model, since it outputs sequences of notes at each state. It could also be viewed as a Hidden Markov Model, since a sequence of output notes could be generated by different sequences of states in the model. We do not make this distinction since it is meaningful only in the context of fitting model parameters using a method such as the forward/backward algorithm. We do not employ this method because it is not efficient for higher-order HMM's.

4 Neural Network for Baseline Emissions

As discussed previously, each pattern, hence each state in the Markov model, corresponds to the same pattern of intervals starting at different baseline pitches and durations. Upon arriving at each state in the Markov model, we need to generate the baseline pitch and duration for the pattern. In previous work [4], we have used a probabilistic emission function that depends only upon the current state. However, this has the disadvantage of not accounting for the pitch and duration of the *previous* pattern. In the present work, we have chosen to learn the distribution of baselines at the current state conditioned on those of the previous state using a neural network. The network topology we used is shown in Fig. 1 c). The inputs to the network are the previous state number and the pitch and duration baselines of the previous state. The outputs are the pitch and duration baselines for the current state. We interpret the values output by the network as probabilities, after normalizing, indicating the probability of selecting the corresponding pitch and duration. Our neural network thus models the conditional distribution on pitch and duration baselines for each state, given the output of the previous state. Note that this implies that our Markov model is first-order.



a) b)

Fig. 2. This figure contains the output of our composition system on: a) the Bach Aria b) the four-voice Bach Air.

We use the standard back-propagation algorithm to train the neural network. To generate training examples, we randomly select a point in the training sequences, and look up the pattern in the suffix tree that begins at that point and the pattern that follows it. Since there may be many pattern nodes on the path from the root to a leaf of the suffix tree, we select one of these nodes probabilistically, with higher probability given to longer paths to encourage the use of longer patterns.

5 Experimental Results

In this section, we give two examples of our experimental results. Both pieces are from J.S. Bach: an "Aria" entitled "Bist Du Bei Mir"; and "Air", from Orchestral Suite Number 3. The Aria is a single voice piece, whereas the Air is a four-voice composition. These pieces were obtained from the Mutopia Project [3], which is an archive of public domain classical music pieces. Due to space limitations, the training sequences are not presented. The results of training our system on these pieces are given in Fig. 2.

There are several observations that may be made on the pieces output by our system. First, the main motif, or theme, in the original pieces occurs multiple times in the output, indicating that our approach is successful in extracting and using the signature patterns of pieces. Secondly, the notes of the piece in part a) are constrained to a much smaller range than those in part b). This occurs because part a) was derived from the Aria which contained only a single voice, whereas part b) is derived from four voices of the Air, one of which is in the bass clef. It is interesting to note that while the output of part b) has a much larger range of notes, our method successfully maintains smooth flow between consecutive notes rather than making large interval jumps. Finally, the pieces respect the common music theory concept of key, in that most of the notes are in the key, with a few accidental notes to create variety.

6 Conclusion

In this paper, we have presented a new approach to algorithmic music composition that learns by example. First, we extract patterns of intervals from musical training sequences. We then build a Markov chain with each state representing a pattern and with transitions representing sequencing of patterns. We use a neural network to determine the baseline pitches and durations to apply to the patterns. We present experimental results that show the viability of this algorithmic composition technique using two Bach pieces.

References

1. Hiller, Lejaren, and Isaacson, Leonard, *Experimental Music: Composition with an Electronic Computer*, McGraw Hill, New York, 1959.
2. Miranda, Eduardo Reck., *Composing Music with Computers*, Focal Press, Burlington, MA, 2001.
3. The Mutopia Project, `www.mutopiaproject.org`.
4. Verbeurgt, Karsten, Dinolfo, Michael, and Fayer, Mikhail, “Extracting Patterns in Music for Composition via Markov Chains”, 17^{th} International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Ottawa, Canada, May 2004.

Exploring Case-Based Bayesian Networks and Bayesian Multi-nets for Classification

Ahmed Hussein and Eugene Santos Jr.

Department of Computer Science and Engineering
University of Connecticut
Storrs, CT 06269-3155
{ahussein,eugene}@engr.uconn.edu

Abstract. Recent work in Bayesian classifiers has shown that a better and more flexible representation of domain knowledge results in better classification accuracy. In previous work [1], we have introduced a new type of Bayesian classifier called *Case-Based Bayesian Network (CBBN)* classifiers. We have shown that CBBNs can capture finer levels of semantics than possible in traditional Bayesian Networks (BNs). Consequently, our empirical comparisons showed that CBBN classifiers have considerably improved classification accuracy over traditional BN classifiers. The basic idea behind our CBBN classifiers is to intelligently partition the training data into semantically sound clusters. A local BN classifier can then be learned from each cluster separately. Bayesian Multi-net (BMN) classifiers also try to improve classification accuracy through a simple partitioning of the data by classes. In this paper, we compare our CBBN classifiers to BMN classifiers. Our experimental results show that CBBN classifiers considerably outperform BMN classifiers.

1 Introduction

Recent work in Bayesian classifiers has shown that better and more flexible representations of the relationships between domain attributes significantly improves classification accuracy. For example, in BN classifiers, the *Tree Augmented Naive-Bayes (TAN)* classifier relaxes the structure of the naive-Bayes classifier by approximating the interactions between the attributes using a tree-like structure. In addition, the *Bayesian network Augmented Naive-Bayes (BAN)* classifier extends TAN by allowing the attributes to form an arbitrary graph, rather than only a tree. Another type of BN classifier that permits even more flexible structures is the General Bayesian Network (GBN) classifier which treats the classification node as an ordinary node and identifies a relevant attribute subset around it determined by its Markov blanket. These classifiers have been shown to outperform naive-Bayes classifier [2,3].

Bayesian Multi-net (BMN) classifiers are a generalized form of the augmented naive structure (i.e., TAN and BAN) in the sense that they allow different relationships among attributes for different values of the class variable. A BMN classifier consists of the prior probability distribution of the class variable C and a set of local networks, each corresponding to a value of C. Two forms of BMN classifiers have been examined. Tree Multi-net (BMN-TAN) [3], a generalization of TAN, where the structure of a local network is restricted to tree-like structure and Bayesian Multi-net (BMN-BAN) [4,3], a generalization of BAN, where a local network can have unrestricted structure. Although the structures of these classifiers are strictly more expressive than traditional BN classifiers (i.e., TAN and BAN), experiments have shown that BMN classifiers perform as well as BN classifiers and that neither approach clearly dominates [3].

In previous work [1], we have introduced a new type of Bayesian classifier called *Case-Based Bayesian Network (CBBN)* classifiers. The basic idea behind CBBN classifiers is to intelligently partition the training data into semantically sound clusters. Each cluster is characterized and discriminated from other clusters by a unique assignment to its most relevant and descriptive attributes. This is called indexing. A local BN classifier can then be learned independently from each cluster conditioned on its index. Such an organization of the domain knowledge allows more precise and more relevant representation of the domain dependency relationships than possible in traditional BNs. We compared CBBN classifiers to BN classifiers with different structures (i.e., naive, TAN, BAN and GBN). Our empirical results showed that CBBN classifiers have considerably improved classification accuracy over BN classifiers.

In this paper, we further explore CBBN classifiers by comparing them to BMN classifiers. Our motivations behind this comparison are as follows: First, BMNs' partitioning of the data is simply restricted to the classes which may not fit the best (i.e., inherent) partitioning of the data and is only useful when the relationships among attributes are very different for different classes. In contrast, our CBBNs relax this restriction by permitting the use of *any appropriate* clustering methodology that discovers the best way to partition the data. Thereby, knowledge can be better represented and more accurate classifiers can be constructed. Second, the indices used by CBBNs provide an attribute selection procedure that is able to discard attributes that are irrelevant to classification. We show that CBBN classifiers outperform BMN classifiers. We also show that while BMN classifiers perform significantly worse than BN classifiers in some cases, our CBBN classifiers perform superior to or competitive with BN classifiers.

2 Our CBBN Approach – An Overview

We use a clustering technique to discover meaningful patterns represented by different clusters of data. Each cluster is then characterized and discriminated from other clusters by a unique assignment to its most relevant and descriptive attributes. This assignment is called an index. As we shall see, these indices provide a natural attribute selection for the CBBN classifiers. Intuitively, each cluster represents a piece of the domain knowledge described by the context of its index. These clusters can also be viewed as a set of conditionally independent cases with each case mapped to an index that describes the context of the knowledge relevant to that case. Because of the independence nature of the cases, the knowledge associated with each case can be represented separately by a BN classifier. This representation of the independent cases implies that the relationships among the corresponding attributes might be different for different cases. Thus, instead of assuming fixed relationships between attributes for the whole domain as in traditional BNs, these relationships can vary according to each different context of each case in the same domain. This conclusion is crucial, since it means that two variables X and Y might be directly dependent ($X \rightarrow Y$) in case C_i and independent in case C_j. Moreover, $X \rightarrow Y$ might occur in case C_i while $X \leftarrow Y$ occurs in case C_j. Even if the relationships in different cases are the same, the parameters that represent the strength of these relationships might be different.

Obviously, CBBNs subsume BMNs in that they exploit the inherent clusters in the data to partition the domain knowledge instead of relying on a simple/restricted partitioning of the data according to the classes which may not fit the natural data clusters and is only useful when the relationships among attributes is very different for different classes. Moreover, our CBBN approach provides a novel procedure for discarding irrelevant attributes. The attributes that constitute an index for a cluster have fixed values for all objects in the cluster. We conclude that these attributes are irrelevant to the classification task in this cluster. Therefore, a BN classifier learned from this cluster can safely exclude these attributes.

3 CBBN Classification Model

Constructing a CBBN classification model consists of the following three phases:

Clustering and Indexing: Let D be a data set described by a set of categorical attributes $A_1, A_2, ..., A_n, C$. A clustering algorithm is used to partition the training data set D into a set of clusters $\mathbf{C} = \{C_1, C_2, \ldots, C_k\}$ characterized by a set of mutually exclusive indices $\mathbf{I} = \{I_1, I_2, \ldots, I_k\}$ respectively. This indexing scheme guarantees at most one mapping per a data object to the set $\mathbf{I}$.

In order to generate such an indexing scheme, we begin by initializing $\mathbf{I}$ (a set of k n-dimension vectors) with "don't care" values for all elements of each vector I_i. For a particular cluster C_i, we compute the probability distribution for each attribute, i.e., the frequencies of its possible values estimated from the data in this cluster. We determine the value of each attribute that has the maximum frequency and assign this value to this attribute in I_i if its frequency exceeds an indexing threshold α. The resulting assignment is then used as a description of the objects in C_i, thus we move all objects that are not covered by I_i from C_i to the outliers cluster. The same procedure is repeated for each cluster. We then visit the outliers cluster to check for possible mappings of its objects back to the indexed clusters. These objects are retrieved from the outliers to be placed in a cluster if the objects are compatible to the cluster's description index.

In order to achieve mutual exclusion between the above assignments, we check each two assignments for the mutual exclusion condition (at least one common attribute is assigned differently). If they do not satisfy this condition, we search for the "don't care" attribute in both assignments that can be assigned differently in both of them such that a minimum number of objects is rejected from both clusters due to the new assignments. We then update the members of all clusters, including the outliers, according to the new mutually exclusive assignments. Finally, to produce the index of each cluster, we simply discard any "don't care" attributes in each cluster's assignment. The algorithms for the above procedure can be found in [1].

Learning: We apply a BN learning algorithm to learn a BN classifier from the data objects in each indexed cluster. This local classifier, B_i where $i \in \{1, 2, ..., k\}$, is defined over a subset of attributes $V_i \subset \mathbf{V} = \{A_1, A_2, \ldots, A_n, C\}$. If $V(I_i)$ is the set of the attributes in I_i then $V_i = \mathbf{V} - V(I_i)$. We also learn a BN classifier, B_o, from the outliers cluster defined over the whole set $\mathbf{V}$.

Testing: We test the newly learned CBBN classification model on the given test data set T. Basically, we try to map each test object $(a_1, a_2, \ldots, a_n)$ in T to an index in $\mathbf{I}$ by comparing the attributes assignment in both of them. We then compute $P(C|a_1, a_2, \ldots, a_n)$ from the local BN classifier characterized by that index and assign to C the value that maximizes P. If an object cannot be mapped to any index in $\mathbf{I}$, we map it to B_o as the default classifier.

4 Experimental Results

4.1 Experiment Settings

In order to compare BMN classifiers to our CBBN classifiers, we have learned three types of BMN classifiers: BMN-TAN, BMN-BAN and BMN-BAN* based on three different learning algorithms: Chow-Liu [5], MDL score [6] and CBL2 [7] respectively [1]. We compare the classification accuracy of these classifiers to their corresponding CBBN classifiers (i.e., CBBN-TAN, CBBN-BAN and CBBN-BAN*).

These classifiers have been learned from each data set in a group of twenty-five benchmark data sets obtained from the UCI machine learning repository at (www.cs.uci.edu). In all data sets, objects with missing attribute values have been removed and numerical attributes have been categorized.

For data clustering in CBBNs, we used the clustering algorithm, *k-modes* [8]. This algorithm requires that the user specify the number of clusters k. In this work, we have determined an acceptable range of k for each data set. More specifically, k can take integer values between $k_{min} = 2$ and k_{max} which is the maximum number of clusters estimated such that each cluster has a number of objects sufficient to learn a BN classifier. We then ran our experiments at three different values of k (k_{min}=2, k_{max}, and $k_{arb} \in]k_{min}, k_{max}[$). For the indexing threshold α, we consider an attribute as a descriptive attribute if its maximum frequency is not less than $\alpha_{min} = 0.7$. We then arbitrarily select α for each data set within the range $[\alpha_{min}, 1]$.

[1] naive and GBN structures are not defined for BMN approach.

Table 1. Classification accuracy for ML, BN, BMN and CBBN classifiers

	Datasets							**ML**		**TAN**			**BAN**			**BAN***		
no.	name	n	c	train	test	k_{arb}	α	C4.5	IB	BN	BMN	CBBN	BN	BMN	CBBN	BN	BMN	CBBN
1	australian	14	2	690	CV-5	3	0.80	85.217	81.739	81.159	81.884	87.391	86.957	86.667	96.232	87.246	87.681	**97.101**
2	breast	10	2	683	CV-5	3	0.80	94.436	96.047	95.900	95.900	95.608	96.633	96.340	**98.682**	96.779	96.486	**98.682**
3	car	6	4	1728	CV-5	4	0.85	69.329	66.204	94.097	94.329	**97.280**	90.451	90.394	96.586	94.039	95.081	96.933
4	chess	36	2	2130	1066	4	0.78	**99.390**	95.028	92.495	92.401	95.872	94.090	96.435	96.998	94.184	93.809	96.717
5	cleve	13	2	296	CV-5	2	0.90	73.986	77.027	79.730	79.392	93.581	79.392	80.068	93.243	82.095	81.757	**95.608**
6	crx	15	2	653	CV-5	3	0.95	86.217	77.489	83.920	84.074	94.334	86.524	86.371	94.793	88.055	89.433	**95.100**
7	diabetes	8	2	768	CV-5	3	0.75	76.172	71.484	75.000	74.349	87.891	75.520	75.911	87.500	77.083	78.776	**92.188**
8	DNA	60	3	2000	1186	4	0.70	92.580	75.801	93.592	94.772	96.374	90.135	91.737	**97.218**	88.533	88.533	96.121
9	flare	10	2	1066	CV-5	3	0.80	82.551	82.833	82.552	81.989	89.587	82.645	82.739	94.090	82.833	81.426	**94.934**
10	german	20	2	1000	CV-5	3	0.82	72.300	69.700	72.200	71.800	91.500	73.200	72.400	92.400	76.800	77.100	**94.800**
11	glass	9	7	214	CV-5	2	0.80	62.241	70.561	68.961	69.159	85.514	70.561	65.421	95.794	71.028	68.692	**96.262**
12	heart	13	2	270	CV-5	2	0.87	80.471	80.000	83.704	83.333	92.593	82.963	83.704	94.815	86.296	87.407	**95.185**
13	led24	24	10	200	3000	2	0.85	65.567	39.433	73.800	73.800	82.967	72.600	70.600	**95.767**	74.100	75.200	94.600
14	liver	6	2	345	CV-5	3	0.88	60.870	64.348	65.217	66.377	79.420	66.957	69.275	94.493	67.246	68.114	**95.072**
15	letter	16	26	15000	5000	10	0.85	77.700	72.800	83.460	84.380	94.920	76.640	80.120	91.440	79.300	81.200	**96.060**
16	mofn-3-7-10	10	2	300	1024	3	0.90	85.449	89.355	91.797	91.602	94.727	86.328	86.523	95.508	88.477	88.379	**96.680**
17	nursery	8	2	8640	4320	6	0.80	68.241	66.157	91.713	92.292	95.255	91.296	90.787	**97.593**	93.079	92.685	97.199
18	pima	8	2	768	CV-5	3	0.75	75.130	68.750	74.870	75.521	86.328	74.740	76.432	92.318	79.297	79.036	**93.359**
19	satimage	36	6	4435	2000	5	0.85	83.100	88.800	77.600	78.000	92.500	80.550	77.150	95.750	84.450	80.000	**96.050**
20	segment	19	7	1540	770	3	0.90	93.506	96.104	85.455	82.857	94.805	91.039	90.260	95.584	90.390	91.169	**96.104**
21	shuttle-small	9	7	3866	1934	5	0.77	99.121	**99.586**	98.914	98.862	97.156	98.914	98.966	98.190	97.208	96.329	96.794
22	soybean-large	35	19	562	CV-5	3	0.85	91.993	90.747	58.363	53.559	71.174	92.349	87.189	**96.263**	92.865	89.502	95.196
23	vehicle	18	4	846	CV-5	3	0.85	69.740	63.830	67.967	66.548	74.470	67.494	64.303	**93.498**	71.631	66.785	87.589
24	vote	16	2	435	CV-5	3	0.80	95.172	94.713	88.966	89.655	94.943	90.115	90.345	94.253	95.632	94.943	**97.241**
25	waveform21	21	3	300	4700	2	0.90	74.787	75.766	75.383	73.830	92.553	77.723	76.809	94.553	78.787	78.191	**94.574**
	best result count							1	1	0	0	1	0	0	5	0	0	**16**

Table 2. BMN vs. BN

BMN→	TAN	BAN	BAN*
% win/BN	44	48	40
% imp/BN	-0.409	-0.467	-0.477

Table 3. CBBN vs. BMN

CBBN→	TAN	BAN	BAN*
% win/BMN	92	96	100
% imp/BMN	12.303	16.268	14.337

Table 4. CBBN-BAN* vs. BMN

CBBN-BAN*	TAN	BAN	BAN*
% win/BMN	96	96	100
% imp/BMN	19.672	16.851	14.337

4.2 Classification Accuarcy

Due to space limitations we only show results at $k = k_{arb}$. Similar results have been obtained at $k = k_{min}$ and at $k = k_{max}$. Our experimental results are presented in Table 1. This table shows the classification accuracy for three different classifier's structures (TAN, BAN and BAN*) built based on three different approaches (BN, BMN and CBBN). In most data sets (21 data sets), BMN classifiers give classification accuracy similar to their corresponding structures in BN classifiers. However, in *letter* data set, BMN classifiers work better than BN classifiers while in the three data sets left (e.g. *soybean-large*) they work considerably worse. A quick inspection of the networks learned in *letter* and *soybean-large* reveals that the dependency assertions among the attributes are very different for different classes in *letter* while almost the same in *soybean-large*. In contrast, CBBN classifiers almost always show considerable improvement over BN classifiers or show similar performance (i.e., classification accuracy). This suggests that while simple partitioning of the data by classes used in BMN classifiers is not effective in many cases, our semantically sound clustering almost always leads to performance improvement. Table 1 also shows that our CBBN classifiers perform considerably better than non-bayesian ML classifiers such as C4.5 and IB classifiers.

Table 2 confirms these results by comparing BMN classifiers to BN classifiers. The results in this table reveals that BMN classifiers do not show an average improvement in accuracy over BN classifiers and do not frequently beat them. Furthermore, we have conducted two comparisons between CBBN classifiers and

Table 5. Construction time in CPU seconds

	data set					TAN		BAN*	
no	name	n	N	c	k_{arb}	BMN	CBBN	BMN	CBBN
1	chess	36	2310	2	4	56	73	66	87
2	DNA	60	2000	3	4	212	374	608	957
3	nursery	8	8640	2	6	20	26	22	33
4	flare	10	1066	2	3	5	10	11	13
5	vote	16	435	2	3	6	14	16	18

BMN classifiers. In the first comparison, Table 3, we measure the average improvement in accuracy of CBBN classifiers over their corresponding BMN classifiers. This table shows that CBBN classifiers considerably outperform BMN classifiers for all structures. In the second comparison, Table 4, we measure the average improvement in accuracy of CBBN-BAN* (the best classifier we have in our CBBN model) over the different structures of BMN classifier. This comparison shows that CBBN-BAN* classifier has a significant average improvement in accuracy over BMN classifier for all structures.

4.3 Time Cost

Table 5 shows the construction time of CBBN and BMN classifiers for five selected data sets. The results show that CBBN is more computationally expensive than BMN. This is due to the time needed to accomplish the intelligent partitioning (clustering and indexing). The time cost for the k-modes clustering algorithm is $O(tnkN)$ where t is the number of iterations for k-modes to converge and N is the size of the training set. Obviously, the clustering time is linear in N and k. However, the indexing process is time-consuming, but it may reduce the time cost of the learning process from each cluster because of the smaller number of attributes involved in learning compared to BMN models. A comparison between the time cost of CBBN and BMN models will depend on the number of clusters k and the number of classes c from which we build the local networks.

5 Conclusions and Future Work

We have compared our CBBN classifiers to BMN classifiers. We have shown that our CBBN classifiers have considerably better classification accuracy than BMN classifiers.

We plan to extend this work in the following direction: We suggested using the k-modes clustering algorithm and ran our experiments with three different values of k and with α arbitrarily selected by the user. Although we obtained good results in all runs, there is no guarantee that these are the best possible results. We would like to come up with a procedure to optimize k and α for the best classification accuracy.

Acknowledgements. This work was supported in part by Air Force Office of Scientific Research Grant No. F49620-03-1-0014.

References

1. Santos, E., Hussein, A.: Case-based bayesian network classifiers. In: Proceedings of the Seventeenth International FLAIRS Conference, AAAI Press (2004)
2. Cheng, J., Greiner, R.: Comparing bayesian network classifiers. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence. (1999)

3. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Machine Learning **29** (1997) 131–161
4. Cheng, J., Greiner, R.: Learning bayesian belief network classifiers: Algorithms and system. In: Proceedings of the Fourteenth Canadian Conference on Artificial Intelligence. (2001)
5. Chow, C.K., Liu, C.: Approximating discrete probability distributions with dependence trees. IEEE Transaction on Information Theory **14** (1968) 462–467
6. Lam, W., Bacchus, F.: Learning bayesian belief networks: An approach based on the mdl principle. Computational Intelligence **10** (1994)
7. Cheng, J., Bell, D., Liu, W.: Learning bayesian networks from data: An efficient approach based on information theory. In: Proceedings of the Sixth ACM International Conference on Information and Knowledge Managment. (1997)
8. Huang, Z.: Extensions to the k-means algorithm for clustering large data sets. Data Mining and Knowledge Discovery **2** (1998) 283–304

Feature Extraction of Handwritten Symbols Using Fuzzy Logic

John A. Fitzgerald, Franz Geiselbrechtinger, and Tahar Kechadi

Department of Computer Science,
University College Dublin,
Belfield, Dublin 4, Ireland.
`john.fitzgerald@ucd.ie`

Abstract. Feature extraction is a process whereby the input data is transformed into a set of features which characterise the input, and which can therefore be used to classify the input. This paper presents a new technique for feature extraction from handwritten symbols. We present a two-phase process. Firstly a pre-processing phase generates a chord vector for each handwritten stroke, thereby eliminating noise and greatly reducing the number of sections of the input which need to be assessed as potential features. Secondly fuzzy rules are used to determine membership values of chord sequences in fuzzy sets corresponding to feature types: Line, C-shape and O-shape. According to these membership values the most likely set of features is determined. Proper selection of the properties required for each feature type, and appropriate timing of application of the fuzzy rules, leads to an effective and efficient feature extraction algorithm.

1 Introduction

Automatic recognition of handwritten symbols often begins with feature extraction [1], whereby those sections of the input which characterise the input are extracted. A good feature extraction method generates similar feature sets for a variety of instances of the same symbol, thereby making the subsequent classification task less difficult. Other approaches to symbol recognition include neural network [2] and k-nearest neighbour [3] methods, both of which require extensive training. The advantage of using the feature extraction approach is that a greater understanding of what is present in the symbol is achieved. By combining feature extraction and fuzzy logic [4], the human perception mechanism is imitated.

In this paper a new feature extraction technique is presented. To recognise symbols using this technique involves a three phase process, consisting of a pre-processing phase, a feature extraction phase, and a classification phase. Only the first two phases will be discussed in this paper. The pre-processing phase involves a new chording technique which transforms each handwritten stroke into a vector of chords. This phase eliminates noise and simplifies the input data so that feature extraction rules can be written in terms of chords. In the feature extraction phase, we distinguish only three feature types: *Line*, *C-shape* and *O-shape*, in the belief that any symbol can be represented by a combination of these features. Fuzzy logic is employed to determine membership values for *substrokes* (sequences of consecutive chords) in fuzzy sets corresponding to these feature types. Based on these membership values the highest quality set of substrokes is chosen as the feature extraction result.

2 Chording

The chording task amounts to partitioning each stroke s into a chord vector $\overrightarrow{C} = \langle c_0, \ldots, c_{n-1} \rangle$, where each chord c_i is a sequence of points ${p_1}^i, \ldots, {p_k}^i$ of s such that the last point of c_i and the first point of c_{i+1} coincide. A partition is a *chording* if the points contained in each section $c = p_1, \ldots, p_k$ approximately form the sector of a circle with the line segment $[p_1, p_k]$ as chord.

Since our goal is to produce a chording with as few chords as possible an iterative solution based on merging chords suggests itself. Starting with the original stroke as coarsest chording the chording algorithm sweeps through the current chording and replaces two successive chords a and b by a new chord if the chords satisfy certain joining rules, which are described below, in addition to the requirement that the change in direction from chord a to b is less than $90\,^{\circ}$.

2.1 Smoothing

Initially a fuzzy *smoothing* rule is used which tests if two successive chords a and b can be seen as a straight line. The decision is based on two properties of chords, *distance* and *curvature*. For a section c we define $dist(c)$ to be the maximal distance from c to the line segment $[c_f, c_l]$, where c_f and c_l are the first and last points of c respectively. We define the curvature $curv(c)$ as $2 * dist(c)/|[c_f, c_l]|$, and we define c to be *straight* if $dist(c) < 1.5 \wedge curv(c) < 0.02$.
The smoothing rule can now be stated simply.

– *Two straight chords a, b can be merged if the combined chord $a + b$ is also straight.*

Smoothing compresses the original stroke data considerably. There is generally a reduction to 25% of the points in the stroke. Since the calculations involved are rather cheap, smoothing is quite effective. Also this simple rule compares favorably to other techniques [5].

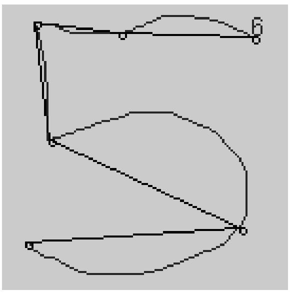
2.2 Shape Formation

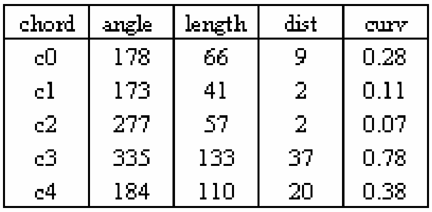
Once smoothing is finished, a *shape formation* rule is applied which tests if two successive chords approximate a sector of a circle. The basic rule is as follows.

– *Join two chords only if merging results in a chord with larger curvature, without distorting the shapes of the original chords.*

The latter is tested as follows. Given two successive chords a and b determine the circle C through a_f, a_l and b_l and calculate distortion factors δ_a and δ_b. The distortion factor, e.g. δ_a, is obtained from $curv(a)$ and the curvature C_a of the circle sector of C for a. If δ_a and δ_b are both within certain bounds, then C is a good approximation of the section formed by chords a and b, and therefore a and b can be merged.

Finally a *finishing* rule merges chords of insignificant length with neighbouring chords.
The chords produced for a '5' and the accompanying chord vector are shown in Fig. 1.
The purposes of chording are to eliminate noise in the original written stroke, to retain only the information which is essential for the feature extraction task, and to greatly reduce the amount of sections of a stroke which need to be evaluated as possible features.





chord	angle	length	dist	curv
c0	178	66	9	0.28
c1	173	41	2	0.11
c2	277	57	2	0.07
c3	335	133	37	0.78
c4	184	110	20	0.38

Fig. 1. The chords produced for a '5'

3 Feature Extraction

After chording takes place, which generates chord vectors $\langle \overrightarrow{C}_0 \ldots, \overrightarrow{C}_v \rangle$ for the strokes $\{s_0, \ldots, s_v\}$ in the symbol, the feature extraction phase occurs. The objective in this phase is to identify the feature set for the symbol. The feature set will be the set of substrokes $F = \{f_0, \ldots, f_{m-1}\}$ encompassing the entire symbol which is of a higher quality than any other possible set of substrokes. Each substroke f_j is a sequence of consecutive chords $(c_a, \ldots, c_b)$ from a chord vector $\overrightarrow{C}_i = \{c_0, \ldots, c_{n-1}\}$, where $0 \leq a \leq b \leq n$ and $0 \leq i \leq v$.

The quality of a set of substrokes, represented by $q(F)$, is dictated by the membership values of the substrokes in F in sets corresponding to feature types. The membership value of a substroke f_j in the set *Line*, for example, is expressed as $\mu_{Line}(f_j)$ or $Line(f_j)$, and represents the confidence that f_j is a line. In the definition of $q(F)$ below, T is one of the fuzzy sets *Line*, *C-shape* or *O-shape*.

$$q(F) = \frac{\sum_{j=0}^{m-1} \mu_T(f_j)}{m}$$

For the symbol in Fig. 1 the feature extraction result is $F = \{(c_0, c_1), (c_2), (c_3, c_4)\}$, where $\mu_{Line}(c_0, c_1) = 0.83$, $\mu_{Line}(c_2) = 0.99$, and $\mu_{Cshape}(c_3, c_4) = 0.94$. It remains to be explained how these membership values are determined using fuzzy rules.

3.1 Fuzzy Rules

The fuzzy rule base contains both *high-level* and *low-level* rules. Membership values in the sets corresponding to feature types are determined by high-level rules. Each high level fuzzy rule defines the *properties* required for a particular feature type, and is of the form:

$$FeatureType(Z) \leftarrow Property_1(Z) \bigcap \ldots \bigcap Property_k(Z).$$

This means the likelihood of a substroke Z being a *FeatureType* is determined by the extent to which $Property_1$ to $Property_k$ are present in Z. In mathematical terms,

$$\mu_{FeatureType}(Z) = \min(\mu_{Property_1}(Z), \ldots, \mu_{Property_k}(Z)).$$

Other norms which could be used rather than the minimum include the Yager intersection connective operator [6] and the weighted generalized means (WGM) operator [7].

Membership values in sets corresponding to properties are determined by *low-level* fuzzy rules. In each low-level rule the fuzzy value $\mu_{Property_i}(Z)$ is defined in terms of values representing various aspects of Z. To express varying degrees of these aspects we use *fuzzy membership functions* such as the S-function [8], Π-shaped function and triangular function [9].

To illustrate the concept of the fuzzy rules, the high-level rule for *Line* is defined below and one of the most basic low-level rules, *Straight*, is described.

$$Line(Z) \leftarrow Straight(Z) \bigcap Smooth(Z) \bigcap Long(Z) \bigcap NoSharpPoints(Z).$$

Straight Rule: The straightness of a substroke depends on how direct a route it takes from its start point to its end point. It is therefore reflected by *startEndRatio(Z)*, which is obtained by dividing the direct distance between the endpoints of Z by the absolute length of Z. If Z is a perfectly straight line, startEndRatio(Z) = 1. If startEndRatio(Z) is less than 0.8, Z is certainly not a straight line. We therefore define *Straight(Z)* as follows:

$$Straight(Z) = S(startEndRatio(Z), 0.8, 0.9, 1).$$

The S-function, so named because of its appearance when plotted, is defined as follows.

$$S(x, a, b, c) = \begin{cases} 0 & \text{if } x \leq a \\ 2(\frac{x-a}{c-a})^2 & \text{if } a \leq x \leq b \\ 1 - 2(\frac{x-c}{c-a})^2 & \text{if } b \leq x \leq c \\ 1 & \text{if } x \geq c \end{cases} \tag{1}$$

Properties required for C-shape and O-shape: C-shapes and O-shapes have certain properties in common. Both are not straight, do not contain major discontinuity points, flexion points or sharp turning points, and are continually curved throughout. In addition to these common properties, a C-shape does not end near to where it started, does not turn through too many degrees, does not intersect itself, and neither endpoint turns in to make the shape resemble '6' or 'e'. An O-shape is round and its start point and end point are close to each other. The fuzzy sets C-shape and O-shape have been defined to reflect these requisite properties.

3.2 Feature Extraction Algorithm

The fuzzy rules form the basis of a feature extraction algorithm which determines the best feature set using numerous efficiency measures. For example, initial detection of the sharp turning points in the symbol can lead to a significant reduction in the number of substrokes to be evaluated, on the basis that such points usually represent boundaries between features.

4 Experimental Results

To measure the performance of the system, 1600 digits and lowercase letters were written by 4 different users, and for each symbol written the system generated the most likely set of features, along with the next most likely sets of features. For each symbol written, the feature set produced as the result was considered a correct result if it was the *expected* feature set.

	1st result	In top 3 results
Expected result:	95.1%	97.8%

For 95.1% of symbols written, the system produced the expected feature set as its first choice result. For a further 2.7% of symbols, the expected feature set was either the second or third most likely result generated by the system. This is significant because if a symbol classifier fails to recognise a symbol using the first choice feature set, the classifier will then attempt to classify the symbol according to the second most likely feature set, and may then be successful.

5 Conclusion

In this paper we have outlined a robust and efficient approach to the task of feature extraction. Through proper selection of the properties required for each feature type, and the use of fuzzy rules which accurately assess the extent to which these properties are present, the likelihood that any substroke is a *Line*, *C-shape* or *O-shape* can be determined with precision and we can subsequently determine the best set of features for the symbol.

For different instances of a particular symbol, this approach will generate feature sets which will all be similar to a prototype feature set for the symbol. Therefore the feature sets produced serve as high quality inputs to a fuzzy classifier, which determines the most likely identity of the symbol based on properties of the features and the relationships between them.

References

1. O. D. Trier, A. K. Jain and T. Taxt, "Feature extraction methods for character recognition - A survey," *Pattern Recognition 29, pp. 641-662*, 1996.
2. I. P. Morns and S. S. Dlay, "The DSFPN: A New Neural Network and Circuit Simulation for Optical Character Recognition," *IEEE Transactions on Signal Processing, Vol. 51, No. 12, pp. 3198-3209*, 2003.
3. I. Soraluze, C. Rodriguez, F. Boto, and A. Perez, "Multidimensional Multistage K-NN Classifiers for Handwritten Digit Recognition," *IWFHR'02, pp. 19-23*, 2002.
4. Ashutosh Malaviya and Liliane Peters, "Fuzzy Feature Description of Handwriting Patterns", *Pattern Recognition, Pergamon Press, Vol. 30, No. 10, pp. 1591-1604*, 1997.
5. Robert Powalka , "Algorithms for on-line cursive script recognition" *Report,The Nottingham Trent University*, 1993.
6. R.R. Yager, "On the representation of multi-agent aggregation using fuzzy logic," *Cybernetics and Systems 21, pp. 575-590*, 1990.
7. G. Klir and T. Folger, "Fuzzy Sets, Uncertainty and Information", *Prentice Hall*, 1991.
8. L.A Zadeh, "Calculus of Fuzzy Restrictions," in *Fuzzy Sets and Their Applications to Cognitive and Decision Processes,* Ed. L.A. Zadeh et al, Academic Press, NY, pp. 1-39, 1975.
9. A. Kaufmann, "Introduction to Theory of Fuzzy Subsets", *Academic Press, NY*, 1975.

Artificial Aging of Faces by Support Vector Machines

Jianning Wang and Charles X. Ling

Department of Computer Science, University of Western Ontario
London, Ontario N6A 5B7, Canada
{jianning, cling}@csd.uwo.ca

Abstract. In this research, we use Support Vector Machines (SVMs) in our system to automatically synthesize the aging effect on human facial images. The coordinates of the facial feature points are used to train the SVMs. Given a new picture, the displacement of the feature points is predicted according to different target ages in the future. The predictions are fed into a warping system to produce the synthesized aged facial images. The results of the prediction using SVMs are analyzed.

1 Introduction and Motivation

After a person has been missing for a number of years, an age-progressed picture of the person is of great help for the law enforcement officials and the general public to more easily identify him/her. So far the job to make these artificially aged facial images is mainly done manually by the forensic artists. On the other hand, since the early 1990's, in the need of a more effective human-machine interaction, the research interest in the area of automatic face recognition has grown significantly. A large number of face identification systems have been developed (Chellapa et al., 1995). Typically, a system for static face recognition is trained using a set of facial features from a number of still training images. The pattern of these features, which make the face of an individual different among others, is thus explicitly or implicitly defined and used for identifying subjects in new images. Many factors can cause the variations in the face images of an individual. When compared with different sources of the intra-individual variation, such as posture and expression, aging displays some unique characteristics, and is the only one that occurs naturally and irreversibly by time. Until recently, there have been a limited number of studies (Pittenger et al., 1979; Burt and Perrett, 1995; Lanitis et al., 2002) attempting to deal with the aging effect on human faces in a systematic way. The approaches proposed previously are based on predefined transformation (Pittenger et al., 1979), statistical processing (Burt and Perrett, 1995), or polynomial approximation and simulation (Lanitis et al., 2002). In this research, we study a more recent machine learning technique, the Support Vector Machine (SVM) method, and apply it in our attempt for a way to automatically synthesize the aging effect on face images.

2 Related Work

The aging process of humans can bring significant change to the face appearance, but until recently, only a few researchers investigated the aging effect on face images in a systematic way. Among the limited number of studies carried out, the approaches that have been proposed can be broadly classified into three categories, i.e. the non-learning (static) methods, the statistical methods and the learning methods. As a static method, the researchers in (Pittenger and Shaw, 1975; Pittenger et al., 1979) employed a coordinate transformation in an attempt to impose age-related changes on human faces. The coordinates of the points on the facial outline are transformed using a group of predefined cardioidal strain functions. Another kind of approaches in artificial face aging is based on statistical methods. In (Burt and Perrett, 1995), the process of aging is investigated using face composites from different age groups and caricature algorithm. The face composites are made by averaging the face shape and texture in a set of images within a certain age range. The differences between the composites for different age groups are evaluated. A caricature algorithm is used to exaggerate those differences that are subsequently applied on the composite or individual images so that their perceived ages are increased. A third kind of face aging methods is based on learning from the aging faces in the training samples. In (Lanitis et al., 2002), the face shape and texture data are extracted for each image in a training set. The principal component analysis (PCA) is applied so that each training face is represented as a weight vector **b**. Using **b** as the parametric feature, two relationships are to be determined: $age = f(\mathbf{b})$ and $\mathbf{b} = f^{-1}(age)$. In their approach, the first one is solved by polynomial approximation learned by a genetic algorithm; the later is prescribed to be in the form of a lookup table which is learned by a simulation procedure. Both of relationships can be learned by using the training images of either each individual or all subjects in the training set. When an unseen face is tested, the aging can follow the global pattern or a combined pattern from the most similar faces in the training set.

3 Data Collection and Preprocessing

For our experiment, we invited a group of volunteers to contribute to our project a set of their photographs at different ages. Some pictures of celebrities have also been used. At least two pictures are from each person. The pictures are required to be taken at least one year apart, in reasonably good lighting condition, in frontal view of the person, and with no major obstruction against the visibility of the face. We pair up the images so that each pair represents the same person at younger (base) and older age. As a result, the image database in our research contains 143 pairs of age progressive face images. Since all pictures were taken in different time and space, to some degree there exist variances in terms of posture, lighting, expression and image quality. This problem should have influence on the result of our experiment. To keep the first stage of our research simple, we study the black and white images only and leave the color ones for future work, so all pictures were converted into black and white ones.

We then normalize the face position by performing affine operations (resize, translation and rotation) on the pictures so that the face shapes in all the images are comparable. In this research, we normalize the face position by fixing the centers of the two pupils in the specific locations in the picture. More precisely, the size of the normalized picture is specified to be 150×188 pixels, and the centers of the left and right pupils are located at the pixel (50, 77) and (100, 77), respectively. For this normalization, the locating of the pupil centers in the original pictures is done manually. The affine transformations are done automatically by a program written in MATLAB script using the built-in Image Processing Toolbox.

The facial features we choose are the coordinates of 58 control points. These control points are the facial landmarks on the jaw, eyes, eyebrows, mouth and nose (see Fig. 1). The automatic detection of these control points is by the active appearance model (AAM) approach (Cootes et al., 2001). The AAM implementation AAM-API (Stegmann, 2002) is used in our experiment. It should be noted that, since the feature extraction by the AAM method may not give the perfect match in some cases, as the last stage, we manually fine-tune the detection result for some pictures.

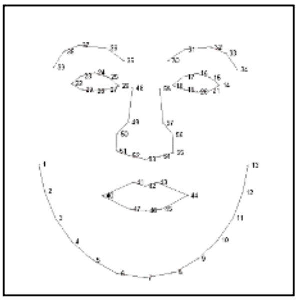


Fig. 1. 58 control points on the frontal

To exclude the influences of the background setting on the visual effect, we isolate the face portion by masking the face-surrounding area in the pictures. The masks for different faces are produced by warping a standard mask to different target shapes detected as described above.

4 Experiments and Results

Given the task of artificial face aging, our proposed solution is to model the alterations caused by aging on human faces using SVMs which are reported to have superior performance in a series of applications including image processing and with the capability of dealing with non-linear regression tasks. At this stage, we focus on studying changes in the face shape. More specifically, we attempt to use SVM regressors to build the relationship between the age progression and the displacement of the 58 feature points we specify in section 3. So far, due to the limitations for both theoretical and practical reasons, we have to treat the change of different feature points as uncorrelated. As a result, we train a different SVM regressor for the target coordinates (x, y) of each of the 58 specified feature points in the age-progressed image. So there are 116 SVM regressors to be trained. The inputs to each SVM include the (x, y) coordinates of the 58 feature points in the younger (base) image, the gender of the subject, the base age, and the progressive age difference. All the inputs are scaled to the range between 0 and 1. Without much domain knowledge in the face aging, we adopt the Gaussian kernel function, which is a popular choice of generic kernels. The SVM implementation we use is LIBSVM 4.0 (Chang and Lin, 2003). Before the training, we use cross validation with the mean squared error criteria to search for the best parameters for each of the SVMs in use. Specifically, three parameters are to be determined for each SVM: the insensitive distance ε, the cost constant C, and the standard deviation σ of the Gaussian kernel function. We search the best values for these parameters in a three dimensional grid by experimenting with each combination of parameters via 5-fold cross validation. After the parameters are determined, the SVMs are trained and used to predict on a set of testing face samples.

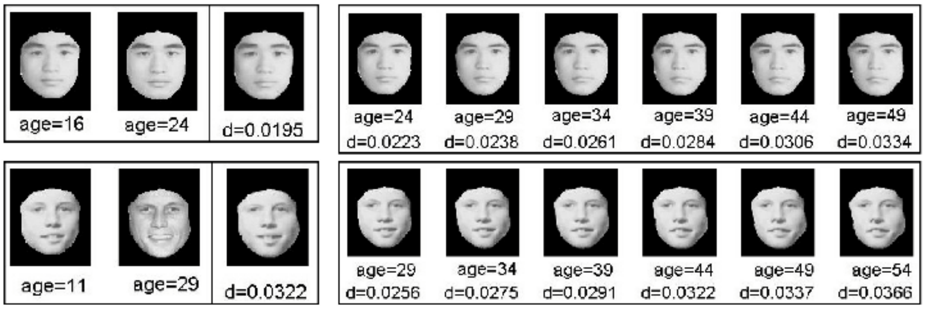


Fig. 2. Examples of the testing faces and the output age-progressed faces. Nine pictures are listed for each of the two subjects, the left column: two real pictures and the real younger picture warped to the real older shape, the right column: synthesized pictures at different specified ages. Parameter d is the Euclidian distance from the target shape to the real younger shape in proportion to the norm of the real younger shape.

For testing purposes, we use a set of face images and specify a number of different progressive ages for each of them. Then we use the trained SVMs to predict the coordinates of facial control points at the target ages. The prediction outputs, together with the base image, are used for image warping to generate the age-progressed images. Fig. 2 shows the examples of the resulting images for two subjects. Note that in these examples the SVMs capture some characteristics about the changes in face shape by aging, e.g. the outline of the face becomes longer, the jaw also becomes longer. By the visual inspection of all the testing results, we can find that in most of the cases the SVMs can predict, for different faces, a few different reasonable changes in shape by aging. This is especially obvious when the base image is the face of a child. The reason may well probably be that a child's face will go through more changes in shape than the adult's. On the other hand, some problems are found through the visual inspection of the results. For example, although the shape changes (by the value of the shape data) in the prediction, the appearance of the face does not change much visually. In a number of cases, when we intend to simulate the aging on a baby face, no matter what shape the system has predicted, it still looks like a baby face after warping. Another problem is that in a few cases the shape is not well continuous so the warped images show a bit distortion.

5 Conclusions and Future Work

Through the evaluation of the experiment results, we can find that, in most of the cases, the SVM model can predict some visually reasonable changes in face shape by aging, although different pictures show such changes in different aspects. For the future work, we shall improve our method by integrating into the system more information such as more control points and texture features about shade and wrinkle, etc. Also, the approach for the regression task of multiple target variables that are correlated should be studied.

References

1. Burt, D.M. and Perrett, D.I. (1995). Perception of Age in Adult Caucasian Male Faces: Computer Graphic Manipulation of Shape and Color Information. In Proc. Royal Soc. London, vol. 259, pp. 137-143.
2. Chang, C.C. and Lin, C.J. (2003) LIBSVM – A Library for Support Vector Machines, Version 2.4. http://www.csie.ntu.edu.tw/~cjlin/libsvm.
3. Chellapa, R., Wilson, C.L., and Sirohey, S. (1995). Human and Machine Recognition of Faces: A Survey. In Proc. IEEE, vol. 83, no.5.
4. Cootes, T.F., Edwards, G.J., and Taylor, C.J. (2001). Active Appearance Models. In IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 23, no. 6, June.
5. Lanitis, A., Taylor, C.J., and Cootes, T.F. (2002). Toward Automatic Simulation of Aging Effects on Fact Images. In IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 24, no. 4, April.
6. Pittenger, J.B. and Shaw, R.E. (1975). Aging Faces as Viscal-Elastic Events: Implications for a Theory of Nonrigid Shape Perception. In Journal of Experimental Psychology: Human Perception and Performance, vol. 1, no. 4, pp. 374-382.
7. Pittenger, J.B., Shaw, R.E., and Mark, L.S. (1979). Perceptual Information for the Age Level of Faces as a Higher Order Invariant of Growth. In Journal of Experimental Psychology: Human Perception and Performance, vol. 5, no. 3, pp. 478-493.
8. Stegmann, M.B. (2002). Analysis and Segmentation of Face Images using Point Annotations and Linear Subspace Techniques. IMM Tech. Rep., Technique University of Denmark, Denmark, Aug.

Solving Dynamic CSPs

Malek Mouhoub

Department of Computer Science, University of Regina
3737 Waskana Parkway, Regina SK, Canada, S4S 0A2
mouhoubm@cs.uregina.ca

Abstract. Constraint Satisfaction problems (CSPs) are a fundamental concept used in many real world applications such as frequency assignment, configuration and conceptual design, scheduling and planning. A main challenge when designing a CSP-based system is the ability to deal with constraints in a dynamic and evolutive environment. During the conceptual phase of design, for example, the designers should be able to add/remove constraints at any time after specifying an initial statement describing the desired properties of a required artifact. We propose in this paper a new dynamic arc consistency algorithm that has a better compromise between time and space than those algorithms proposed in the literature, in addition to the simplicity of its implementation. Experimental tests on randomly generated CSPs demonstrate the efficiency of our algorithm to deal with large size problems in a dynamic environment.

Keywords: Constraint Satisfaction, Dynamic Arc Consistency.

1 Introduction

A Constraint Satisfaction problem (CSP) [1,2,3] involves a list of variables defined on finite domains of values and a list of relations restricting the values that the variables can take. If the relations are binary we talk about binary CSPs. CSPs are a fundamental concept used in many real world applications such as engineering conceptual design, frequency assignment, scheduling, planning and molecular biology. A main challenge when designing a CSP-based system, for the above applications and others, is the ability to deal, in an efficient way, with constraints in a dynamic and evolutive environment. One example, in the area of engineering conceptual design, is when the designers add/remove constraints after specifying an initial statement describing the desired properties of a required artifact during the conceptual phase of design. This motivates us to adapt the current constraint resolution techniques in order to process the constraints in an incremental way. Solving a CSP consists of finding an assignment of values to each variable such that all relations (or constraints) are satisfied. A CSP is known to be an NP-Hard problem. Indeed, looking for a possible solution to a CSP requires a backtrack search algorithm of exponential complexity in time[1].

[1] Note that some CSP problems can be solved in polynomial time. For example, if the constraint graph corresponding to the CSP has no loops, then the CSP can be solved in $O(nd^2)$ where n is the number of variables of the problem and d is the domain size of the different variables

To overcome this difficulty in practice, local consistency techniques are used in a pre-processing phase to reduce the size of the search space before the backtrack search procedure. A k-consistency algorithm removes all inconsistencies involving all subsets of k variables belonging to N. The k-consistency problem is polynomial in time, $O(N^k)$, where N is the number of variables. A k-consistency algorithm does not solve the constraint satisfaction problem, but simplifies it. Due to the incompleteness of constraint propagation, in the general case, search is necessary to solve the CSP problem, even to check if a single solution exists. Note that local consistency can also be used during the search, following the forward check strategy, in order to prevent earlier later failures [2]. When $k = 2$ we talk about arc consistency. An arc consistency algorithm transforms the network of constraints into an equivalent and simpler one by removing, from the domain of each variable, some values that cannot belong to any global solution.

We propose in this paper a new dynamic arc consistency algorithm that has a better compromise between time and space than those algorithms proposed in the literature [4,5,6], in addition to the simplicity of its implementation. In order to evaluate the performance in time cost of the techniques we propose, experimental tests on randomly generated CSPs have been performed. The results demonstrate the efficiency of our algorithm to deal with large size dynamic CSPs.

The rest of the paper is organized as follows. In the next section we will present the dynamic arc consistency algorithms proposed in the literature and the new dynamic arc consistency algorithm we propose. Theoretical comparison of our algorithm and those proposed in the literature is covered in this section as well. The experimental part of our work is presented in section 3. Finally, concluding remarks and possible perspectives are listed in section 4.

2 Dynamic Arc Consistency: The Algorithm AC3.1|DC

The arc-consistency algorithms proposed in the literature[1,7,8,9] can easily be adapted to update the variable domains incrementally when adding a new constraint. This simply consists of performing the arc consistency between the variables sharing the new constraint and propagate the change to the rest of the constraint network. However, the way the arc consistency algorithm has to proceed with constraint relaxation is more complex. Indeed, when a constraint is retracted the algorithm should be able to put back those values removed because of the relaxed constraint and propagate this change to the entire graph. Thus, traditional arc consistency algorithms have to be modified so that it will be possible to find those values which need to be restored anytime a constraint is relaxed. Bessière has proposed DnAC-4[4] which is an adaptation of AC-4[7] to deal with constraint relaxations. This algorithm stores a justification for each deleted value. These justifications are then used to determine the set of values that have been removed because of the relaxed constraint and so can process relaxations incrementally. DnAC-4 inherits the bad time and space complexity of AC-4. Indeed, comparing to AC-3 for example, AC-4 has a bad average time complexity[10]. The worst-case space complexity of DnAC-4 is $O(ed^2 + nd)$ (e, d and n are respectively the number of constraints, the domain size of the variables and the number of variables). To work out the drawback of AC-4 while keeping an optimal worst case complexity, Bessière has proposed AC-6[8]. Debruyne has then proposed DnAC-6 adapting the idea of AC-6 for dynamic CSPs by using justifications similar to those of DnAC-4[5]. While keeping an optimal worst case time complexity ($O(ed^2)$), DnAC-6 has a lower space requirements ($O(ed + nd)$) than DnAC-4. To solve the problem of space complexity, Neveu and Berlandier proposed AC|DC[11]. AC|DC is based on AC-3 and does not require data structures for storing justifications. Thus it has a very good space complexity ($O(e + nd)$) but is less efficient in time than DnAC-4. Indeed, with its $O(ed^3)$ worst case time complexity, it is not the algorithm of choice for large dynamic CSPs. Our goal here is to develop an algorithm that has a better compromise between running time and memory space than the above three algorithms. More precisely, our ambition is to have an algorithm with the $O(ed^2)$ worst case time complexity of DnAC-6 but without the need of using complex and space expensive data structures to store the justifications. More recently Zhang and Yap proposed an new version of AC-3 (called AC-3.1) achieving the optimal worst case time complexity with $O(ed^2)$ ([12]). We have then decided to adapt this new algorithm, in the same way as it was done for AC|DC, in order to deal with constraint relaxations. More precisely, our idea is to integrate the AC-3.1 into the AC|DC algorithm since that algorithm is based on AC-3. The problem with the AC|DC algorithm is that it relies solely on the AC-3 algorithm and did not keep support lists like DnAC4 or DnAC6 causing the relaxation of a constraint to be fairly time consuming. This is also the reason for its worst case time complexity of $O(ed^3)$. If AC-3.1 is integrated into the AC|DC algorithm, then by theory the worst case time complexity should be $O(ed^2)$. In addition to this, the worst case space complexity remains the same as the original AC|DC algorithm of $O(e + nd)$. During preliminary tests performed on randomly generated CSPs, we concluded that this new adaptation of AC|DC using the AC-3.1 algorithm improves the time complexity when concerning restrictions as can be concluded by the results shown by Zhang and Yap [12]. The more interesting question is whether this algorithm's time complexity remains the same during retractions. The way AC3.1|DC deals with relaxations is as follows. For any retracted constraint (k, m) between the variables k and m, we perform the following three phases:

1. An estimation (over-estimation) of the set of values that have been removed because of the constraint (k, m) is first determined by looking for the values removed from the domains of k and m that have no support on (k, m). Indeed, those values already suppressed from the domain of k (resp m) and which do have a support on (k, m), do not need to be put back since they have been suppressed because of another constraint.
2. The above set is then propagated to the other variables. In this phase, for each value (k, a) (resp (m, b)) added to the domain of k (resp m) we will look for those values removed from the domain of the variables adjacent to k (resp

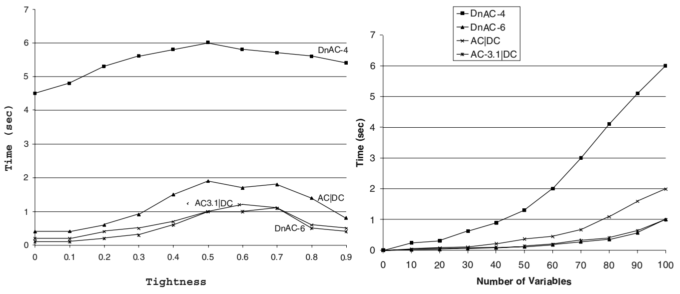


Fig. 1. Comparative tests of the dynamic arc-consistency algorithms

m) supported by (k, a) (resp (m, b)). These values will then be propagated to the adjacent variables.

3. Finally a filtering procedure based on AC-3.1 is then performed to remove from the estimated set the values which are not arc-consistent with respect to the relaxed problem.

The worst case time complexity of the first phase is $O(d^2)$. AC-3.1 is applied in the third phase and thus the complexity is $O(ed^2)$. Since the values are propagated only once then the complexity of the second phase is also $O(ed^2)$. Thus the overall complexity of the relaxation is $O(ed^2)$. In terms of space complexity, AC-3.1 requires an array storing the resume point for each variable value (in order to have $O(ed^2)$ time complexity). The space required by this array is $O(nd)$. If we add to this the $O(e + nd)$ space requirement of the traditional AC-3 algorithm, the overall space requirement is $O(e + nd)$. A theoretical comparison of the four dynamic arc consistency algorithms is summarized in the following table. As we can see, AC-3.1|DC has a better compromise between time and space costs.

	DnAC-4	DnAC-6	AC|DC	AC-3.1|DC
Space complexity	$O(ed^2 + nd)$	$O(ed + nd)$	$O(e + nd)$	$O(e + nd)$
Time complexity	$O(ed^2)$	$O(ed^2)$	$O(ed^3)$	$O(ed^2)$

3 Experimentation

Theoretical comparison of the four dynamic arc consistency algorithms shows that AC3.1|DC has a better compromise between time and space costs than the other three algorithms. In order to see if the same affirmation can be concluded in practice we have performed comparative tests on dynamic CSPs randomly generated as we will show in the following. The experiments are performed on a Sun Sparc 10 and all procedures are coded in C|C++. Given n the number of variables and d the domain size, each CSP instance is randomly generated as follows. We first generate n sets of d natural numbers. $\frac{n(n-1)}{2}$ constraints are then picked randomly from a set of arithmetic relations $\{=, \neq, <, \leq, >, \geq, \ldots\}$. The left chart of figure 1 shows the performance in time of each arc consistency algorithm when achieving the arc consistency in a dynamic environment, as follows. Starting from a CSP having $n = 100$, $d = 50$ and 0 constraints, restrictions are obtained by adding relations randomly generated as shown in the previous subsection until a complete graph (number of constraints=$\frac{n(n-1)}{2}$) is obtained. Afterwards, relaxations are performed until the graph is 50% constrained (number of constraints=$\frac{n(n-1)}{4}$). As we can easily see, the results provided by AC-3.1|DC are better than that of AC|DC and DnAC-4 in all cases. Also AC-3.1|DC algorithm is comparable to DnAC6 (that has the best running time of the three dynamic arc consistency algorithms) as can be seen in the left chart of figure 1. Using the same way as above, we have performed comparative tests on randomly generated DCSPs, each having a different number of variables. The domain size is fixed to 50. The results presented in the right chart of figure 1 demonstrate the efficiency of both AC-3.1|DC and DnAC-6 over the other two algorithms.

4 Conclusion and Future Work

In this paper we have presented new algorithms for maintaining the consistency of a CSP in a dynamic environment at the arc consistency level. Comparing to the dynamic arc consistency algorithms proposed in the literature, ours offers a better compromise between time and space costs.

One perspective of our work is to handle the relaxation of constraints during the backtrack search phase. For instance, suppose that during the backtrack search process a new constraint is removed. In this case, would it be worthwhile to reconsider the assignment of the variables already instantiated or would it be more costly than just continuing with the backtrack search. If, for example, the suppression of the constraint leads to a constraint network without loop then the CSP can easily be solved with a polynomial time algorithm. Another perspective is to use the dynamic methods we propose for solving conditional CSPs. Conditional CSPs are CSPs containing variables whose existence depends on the values chosen for other variables. In this case we will have to maintain the consistency of the CSP any time new variables are added.

References

1. Mackworth, A.K.: Consistency in networks of relations. Artificial Intelligence **8** (1977) 99–118
2. Haralick, R., Elliott, G.: Increasing tree search efficiency for Constraint Satisfaction Problems. Artificial Intelligence **14** (1980) 263–313
3. Kumar, V.: Algorithms for constraint satisfaction problems: A survey. AI Magazine **13** (1992) 32–44
4. Bessière, C.: Arc-consistency in dynamic constraint satisfaction problems. In: AAAI'91, Anaheim, CA (1991) 221–226
5. Debruyne, R.: Les algorithmes d'arc-consistance dans les csp dynamiques. Revue d'Intelligence Artificielle **9** (1995) 239–267

6. Neuveu, B., Berlandier, P.: Maintaining arc consistency through constraint retraction. In: ICTAI'94. (1994) 426–431
7. Mohr, R., Henderson, T.: Arc and path consistency revisited. Artificial Intelligence **28** (1986) 225–233
8. Bessière, C.: Arc-consistency and arc-consistency again. Artificial Intelligence **65** (1994) 179–190
9. Bessière, C., Freuder, E., Regin, J.: Using inference to reduce arc consistency computation. In: IJCAI'95, Montréal, Canada (1995) 592–598
10. Wallace, R.J.: Why AC-3 is almost always better than AC-4 for establishing arc consistency in CSPs. In: IJCAI'93, Chambery, France (1993) 239–245
11. Neuveu, B., Berlandier, P.: Arc-consistency for dynamic constraint satisfaction problems : An rms free approach. In: ECAI-94, Workshop on Constraint Satisfaction Issues Raised by Practical Applications, Amsterdam (1994)
12. Zhang, Y., Yap, R.H.C.: Making ac-3 an optimal algorithm. In: Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01), Seattle, WA (2001) 316–321

Histogram Arc Consistency as a Value Ordering Heuristic

Wei Liu and William S. Havens

Intelligent Systems Laboratory, School of Computing Science, Simon Fraser University Burnaby, BC, Canada, V5A 1S6
{wliua, havens}@cs.sfu.ca
http://www.cs.sfu.ca/research/groups/ISL/index.html

Abstract. The Constraint Satisfaction Problem (CSP) is NP-hard. Finding solutions requires searching in an exponential space of possible variable assignments. Good value ordering heuristics are essential for finding solutions to CSPs. Such heuristics estimate the marginal probability that any particular variable assignment will appear in a globally consistent solution. Unfortunately, computing such solution probabilities exactly is also NP-hard. Thus estimation algorithms are required. Previous results have been very encouraging but computationally expensive. In this paper, we present two new algorithms, called Histogram Arc Consistency (HAC) and μ Arc Consistency(μAC), which generate fast estimates of solution probabilities during constraint propagation. This information is used as value ordering heuristics to guide backtrack search. Our experimental results on random CSPs show that these methods indeed provide significant heuristic guidance compared to previous methods while remaining efficient to compute.

1 Introduction

Heuristics have been successfully applied in search algorithms to solve Constraint Satisfaction Problems (CSP). An efficient value ordering heuristic is to choose a value which appears in most solutions, or "counting the number of solutions". However, this information is only available after all solutions have been found.

Various researches have been done on generating approximations of solution counts [6] [1] [4] [2]. In this paper, we present two new algorithms, called Histogram Arc Consistency(HAC) algorithm, and μ Arc Consistency (μAC) algorithm. These algorithms approximate the solution counts by propagating the number of constraints that are satisfiable. Local information is used to determine global properties. We aware that the solution counts obtained by our algorithms are not exact, but we argue that the exact solution counts are not necessary when they are used as value ordering heuristics. Our very coarse approximation is good enough to guide the search algorithm effectively.

Proposed algorithms are tested using randomly generated CSPs. Numbers of backtracks used to find first solutions and CPU times are collected. These algorithms are compared with the traditional arc consistency algorithm [5] and with the pAC algorithm [4]. Experimental results show that our proposed algorithms provide valuable heuristic information. When HAC and μAC are used as value ordering heuristics to guide the search algorithm, fewer backtracks are required to find solutions than the traditional arc consistency algorithm. Our experimental results also show that our methods are quick to compute. The right balance is established between heuristic guidance, stability and efficiency.

2 Histogram Arc Consistency (HAC)

The Histogram Arc Consistency algorithm is a natural extension of both the traditional arc consistency algorithm [5] and the probabilistic arc consistency algorithm [4]. In the traditional arc consistency algorithm, each domain value is associated with a constraint value representing the membership status of this value. This constraint value is of type boolean which tells us whether the associated domain value is in the live domain or not. Constraints are combined using boolean operations, $\vee$ and $\wedge$. One of the problems with this algorithm is that it only distinguishes domain values that are in the live domain from those that are not, it does not give these values any likelihood of being in a consistent global solution. In the HAC algorithm, we change the data type of the constraint value from boolean to integer, and we change the operations from boolean operations to integer operations, *i.e.* $+$ and $\times$. The Histogram Arc Consistency is achieved by updating constraint valuations using the following updating formula incrementally until no element is deleted from variable's live domain.

$$\forall v_i \in D_{x_i}, f(v_i) = \prod_{x_j \in N(x_i)} \sum_{v_j \in D_{x_j}} C_{x_i,x_j}(v_i, v_j) m(v_j)$$

where

- D_{x_i} is the current live domain of variable x_i.
- $f(v_i)$ is the HAC valuation for value v_i in x_i.
- $\prod$ represents integer product.
- $\sum$ represents integer sum.
- $N(x_i)$ represents neighboring variables of x_i.
- $C_{x_i,x_j}(v_i, v_j)$ is 1 if the pair $\{x_i = v_i, x_j = v_j\}$ is allowed by the constraint C_{x_i,x_j}, and 0 otherwise.
- $m(v_j)$ is the membership of v_j in the live domain of x_j. It is obtained by the following formula
$$m(v_j) = \begin{cases} 1 & f(v_j) > 0 \\ 0 & f(v_j) = 0 \end{cases}$$

In the above function, $f(v_i)$ is the constraint valuation associated with the assignment $< x_i, v_i >$. The additive operation tells us how many domain values in the neighboring variable x_j supporting the assignment $< x_i, v_i >$. The number of supports from different neighbors are combined using integer product operation, $\prod$. A value that gets the support count of 0 means that there is no support from its neighboring variables, and thus can be removed from variable's live domain. Removing domain values will cause constraint values in their neighboring variables to changed. To achieve Histogram Arc Consistency for the whole problem, these constraint valuations are updated incrementally until no values are deleted from any variable live domain.

The HAC-REVISE(x_i, x_j) procedure shown in Figure 1 revises constraint valuations for variable x_i based on the live domain of variable x_j. In this procedure, m_i is a vector containing the membership of domain values of variable x_i, as defined above. s_{ij} is a vector containing the number of supports for domain values in variable x_i from x_j. Initially, all constraint values are set to 1. At lines 2, and 3, previous memberships of domain values for variable x_i, and previous supports for domain values of x_i from neighboring variable x_j are saved into vector $\hat{m_i}$ and $\hat{s_{ij}}$ respectively. For each value v_i in the live domain of variable x_i, line 8 counts the new number of supports based on the current live domain of its neighboring variable x_j. Line 10 combines the supports for value v_i from different neighboring variables using multiplication. To avoid the double counting problem, the old number of supports from variable x_j, $\hat{s_{ij}}$, is deleted from the updated support valuation. The HAC-REVISE procedure returns true if the live domain of variable x_i is changed. This change is propagated through the constraint network.

```
precedure HAC-REVISE($x_i, x_j$)
1.  begin
2.      $\hat{m_i} \leftarrow m_i$
3.      $\hat{s_{ij}} \leftarrow s_{ij}$
4.      $s_{ij} \leftarrow \mathbf{0}$
5.      for $v_i \leftarrow$ each element in $D_{x_i}$ do
6.          for $v_j \leftarrow$ each element in $D_{x_j}$ do
7.              if $(C_{x_i,x_j}(v_i, v_j) \wedge m_j[v_j])$ then
8.                  $s_{ij}[v_i]$ ++;
9.      for each element $k$ in $f_i$ do
10.         $f_i[k] = f_i[k] * s_{ij}[k] / \hat{s_{ij}}[k]$
11.     return $(m_i \neq \hat{m_i})$
12. end
```

Fig. 1. The HAC-REVISE algorithm

The HAC algorithm can be combined with the backtrack search algorithm to solve CSPs. It can be used in preprocessing before the search algorithm to reduce the size of the search tree or used during search (after each assignment is made) to prune off search space. In addition, the HAC algorithm provides heuristic order for the domain values, i.e., choose the value in the variable domain with the largest constraint valuation.

3 μ Arc Consistency (μAC)

In most cases, we are more interested in finding a solution quickly than the completeness of the algorithm. So the idea behind the μAC algorithm is: We aggressively reduce variable live domains by simply removing domain values with small HAC valuations from variable live domains. We quickly search through this very small search space. If no solution is found, we expand the search space a little more by putting more values back to live domains and search again. If eventually we have to put all values back into variable live domains and search, the μAC algorithm exhausts the entire search space. Thus μAC is guaranteed to find the solution if one exists. The worst case complexity of the μAC algorithm is $O(n + 2^n + 3^n + ... + d^n) = O(d^n)$. However, as pointed out in [3], if the heuristic is good, i.e., if HAC algorithm provides a good ordering of domain values, the worst scenario is unlikely to happen.

Basically, μAC is HAC with a μ parameter. In HAC, when any of the HAC valuations becomes zero the associated domain value is deleted from the variable's live domain and all affected constraints are reexamined. In μAC, 0 is replaced by μ. So if any HAC valuation becomes less than μ, the associated domain value is deleted from the variable's live domain and related constraints are reexamined. By aggressively reducing the size of the search tree, hopefully, we can find a solution quickly and with less backtracks.

4 Experimental Results

Proposed algorithms are tested using randomly generated CSPs. Testing problems adapted here have 20 variables (n), and each variable has domain size 10 (m). The probability that a constraint exists between any pair of variables, represented as p_1, is set to 0.4. The probability that a given pair of values is disallowed, called p_2, ranges from 0.01 to 1.0 and increments by 0.01. For each pair of p_1 and p_2, 200 different CSP instances are tested.

These test problems are solved using four different algorithms. They are: (1). backtrack search with traditional arc consistency (Full Look-ahead); (2). backtrack search with HAC as a value ordering heuristic (HAC Look-ahead); (3). backtrack search with μAC as a value ordering heuristic (μAC Look-ahead) and (4). backtrack search with pAC as a value ordering heuristic (pAC Look-ahead). For each test problem, we record the number of backtracks required to find first solutions, and also CPU times taken by these algorithms.

In Figure 2, 7000 problems with $n = 20, m = 10, p_1 = 0.4, p_2 \in [0.2, 0.54]$ are selected. The figure plots total numbers of backtracks used by each algorithm to find first solutions against constraint tightness (p_2). On average, HAC saved 22.21% of backtracks from the traditional arc consistency algorithm, μAC saved 44.74% and pAC saved 50.15%.

We also compared CPU times taken to find first solutions for these problems. As shown in Figure 3, the pAC Look-ahead algorithm takes much longer CPU time than HAC Look-ahead and μAC Look-ahead algorithms. Due to the overhead of integer operations and the extra search steps to find the most promising value in a variable domain, HAC Look-ahead and μAC Look-ahead algorithms take longer CPU time than the Full Look-ahead algorithm.

To test the performances of HAC and μAC for larger CSP problems, we test a set of problems with $n = 60, m = 30, p_1 = 0.3$ and $p_2 \in [0.1, 0.3]$. As shown in Figure 4, the pAC algorithm takes a large amount of CPU time to find solutions. HAC Look-ahead and μAC Look-ahead algorithms take less CPU time than the Full Look-ahead algorithm to find the first solutions. The reduced search cost compensates the overhead incurred in the proposed algorithms.

5 Conclusions and Future Works

In this thesis, we study the practical use of generalized propagation techniques. Heuristics generated by our algorithms, HAC and μAC, are derived from arc consistency operations directly.

When comparing HAC with previous algorithms which approximate the solution counts on simplified CSPs [1,6,7], a major difference is that those algorithms ignore loops in the constraint network and assume single connection or independence among subproblems. The deletion of domain value is not propagated through the constraint network. Resulting marginal solution counts are not locally consistent with their neighboring variables. Our algorithms maintain the original structure of the constraint graph. Deletions of locally inconsistent domain values are propagated through the constraint network while arc consistency is maintained throughout the whole process. The pAC algorithm does not ignore loops in the constraint graph. However, it tries to generate solution probabilities as accurate as possible. The pAC algorithm not only propagates the deletion of domain value, it also propagates solution probabilities of these domain values until they do not change. Any small changes in the solution probabilities will trigger a mass constraint propagation. This causes an over counting problem and convergence problem. Our algorithms instead compute a quick estimation. We propagate the solution counts until no value is deleted from any variable domain thus guaranteeing convergence.

Experimental results show that when used as value ordering heuristics, HAC and μAC algorithms performed almost as good as the pAC algorithm, however, take much less CPU time. The right balance is established between heuristic guidance, stability and efficiency.

HAC is similar to Geelen's value ordering heuristic [2]. The major difference is on their implementations. Because updating the solution counts incrementally during the process of achieving arc consistency gets the same results as recalculating supports whenever the domain of a neighboring variable is changed. The μAC algorithm takes one step further. It assumes accuracy of the marginal solution counts generated by HAC and removes all domain values with small valuations from variable live domains. These forced domain reductions are propagated through the constraint network. Experimental results show that, although the approximated solution counts generated by the HAC algorithm provide good

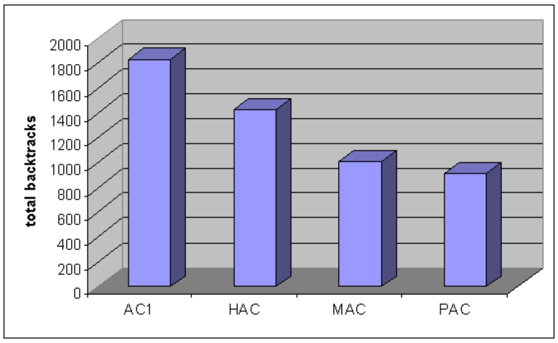


Fig. 2. Total Number of Backtracks Used to Find the First Solution for problem with $n = 20, m = 10, p_1 = 0.4, p_2 \in [0.2, 0.54]$

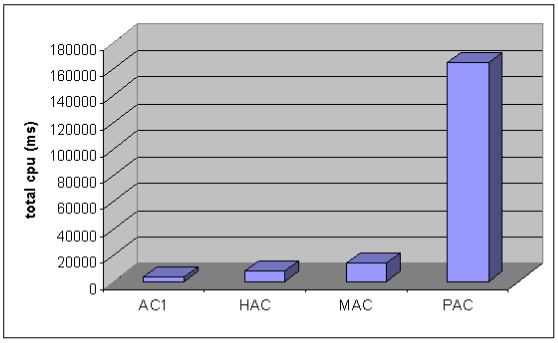


Fig. 3. Total CPU Taken to Find the First Solution for problem with $n = 20, m = 10, p_1 = 0.4, p_2 \in [0.2, 0.54]$

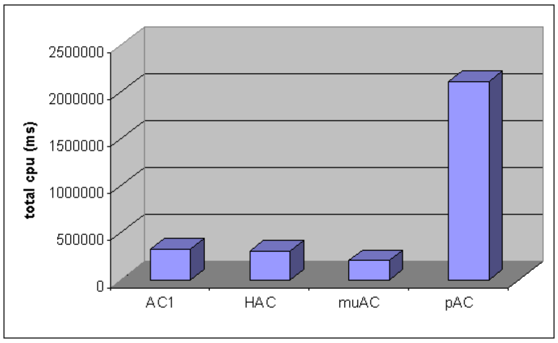


Fig. 4. Total CPU taken to find first solutions for problems with $n = 60, m = 30, p_1 = 0.3, p_2 \in [0.1, 0.3]$

heuristic guidance for the search algorithm, it is the aggressive value pruning of the μAC algorithm that really improves the performance.

For future research directions, we suggest the following: 1). explore other generalized propagation schemes and combination operators for estimating the solution counts. 2). further investigate the performances of HAC and μAC algorithms using real-world problems.

References

1. R. Dechter and J. Pearl. Network-based heuristics for constraint-satisfaction problems. *Artificial Intelligence*, 34:1–34, 1988.
2. P. A. Geelen. Dual viewpoint heuristics for binary constraint satisfaction problems. In *Proceedings of the 10th European Conference on Artificial Intelligence*, pages 31–35, 1992.
3. W. D. Harvey and M. L. Ginsberg. Limited discrepancy search. In *Proc. of the Fourteen International Joint Conference on Artificial Intelligence(IJCAI-95)*, pages 607–615, 1995.
4. M. C. Horsch, W. S. Havens, and A. Ghose. Generalized arc consistency with application to maxcsp. In R. Cohen and B. Spencer, editors, *Proceedings of Canadian Conference on AI*, pages 104–118, 2002.
5. A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.
6. A. Meisels, S. E. Shimonoy, and G. Solotorevsky. Bayes networks for estimating the number of solutions to a csp. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 1–34, 1997.
7. M. Vernooy and W. S. Havens. An examination of probabilistic value-ordering heuristics. In *Proceedings of the 12th Australian Joint Conference on Artificial Intelligence*, 1999.

Knowledge Provenance

Mark S. Fox and Jingwei Huang

Enterprise Integration Laboratory, University of Toronto
40 St. George Street, Toronto, ON M5S 3G8, Canada
{msf,jingwei}@eil.utoronto.ca

Abstract. Knowledge Provenance is proposed to address the problem about how to determine the validity and origin of information/knowledge on the web by means of modeling and maintaining information source and dependency, as well as trust structure. Four levels of Knowledge Provenance are introduced: Static, where the validity of knowledge does not change over time; Dynamic, where validity may change over time; Uncertain, where the truth values and trust relation are uncertain; and Judgmental: where the societal processes for determining certainty of knowledge are defined. An ontology, semantics and implementation using RDFS is provided for Static Knowledge Provenance.

1 Introduction

In this paper, Knowledge Provenance (KP) is proposed to address the problem about how to determine the origin and validity of information/knowledge on the web. The problem arises from many directions: information may no longer be relevant (e.g., discontinued products or old operating procedures), may contain incorrect information (e.g., news stories), and may even be outright lies. For example, in 1999, two men posted fraudulent corporate information on electronic bulletin boards, which caused the stock price of a company (NEI) to soar from $0.13 to $15, resulting in their making a profit of more than $350,000 [Mayorkas, 2000].

Interest in addressing the issue of web information trustworthiness has appeared under the umbrella of the "Web of Trust" that is identified as the top layer of The Semantic Web (see [Berners-Lee 2003] slide 26, 27). No doubt, digital signature and digital certification ([Simon et al, 2001]) will play important roles in "Web of Trust". However, they only provide an approach to certify an individual's identification and information integrity, but they do not determine whether this individual could be trusted. Trustworthiness of the individual is supposed to be evaluated by each application. For the purpose of secure web access control, Blaze et al, (1996) first introduced "decentralized trust management" to separate trust management from applications. Since then, trust management has been developed as an important field (Khare & Rifkin 1997). More general trust management models in the context of web applications are being developed. For example, Golbeck et al. (2002) study trust propagation in social networks. Although tightly related, in the context of knowledge provenance, trust management only considers direct and indirect trust relationships to information creators but does not consider the dependency among information. KP will address both trust relationships and the dependency among information.

KP aims at creating an approach for both people and software agents to determine the origin and validity of web knowledge by means of modeling and maintaining information source and dependency, as well as trust structure. The major questions that need to be answered in KP are: For any piece of web information, where does it come from? Who created it? Can it be believed? The technologies developed in AI, such as Truth Maintenance System, temporal logic, uncertainty logic, etc., provide basic approach for knowledge representation and reasoning in knowledge provenance.

Philosophically, we believe the web will always be a morass of uncertain and incomplete information. But we also believe that it is possible to annotate web content to create islands of certainty. Towards this end, Knowledge Provenance introduces 4 levels of provenance that range from strong provenance (corresponding to high certainty) to weak provenance (corresponding to high uncertainty). Level 1 (Static KP) develops the fundamental concepts for KP, and focuses on provenance of static and certain information; Level 2 (Dynamic KP) considers how the validity of information may change over time; Level 3 (Uncertainty-oriented KP) considers uncertain truth value and uncertain trust relations; Level 4 (Judgment-based KP) focuses on social processes necessary to support provenance. Since Static KP is the foundation to develop other levels of KP. This paper focuses on Static KP to introduce the fundamental concepts and the basic approach of KP.

This paper is organized as follows: Section 2 introduces the basic concepts of KP and provides motivating scenarios for static KP. Section 3 introduces Static KP model. Section 4 illustrates how to use KP, which includes annotating web documents and provenance reasoning. Section 5 gives a summary and a view on future work.

2 What Is Knowledge Provenance?

The basic unit of web information to be considered in KP is a "proposition". A proposition, as defined in First Order Logic, is a declarative sentence that is either true or false. A proposition is the smallest piece of information to which provenance-related attributes may be ascribed. An information creator may define a phrase (logically being a proposition), a sentence, a paragraph, even a whole document as a proposition. Not only text but also an xml element could be defined as a proposition.

To use KP, information creators need to annotate web documents with KP metadata to describe the provenance-related attributes, such as who is proposition creator and what is the premise proposition which this proposition depends on. A software plugged in web browsers is expected to assist information creators to annotate their web documents; information users (provenance requesters) need define their personalized trust relations to tell whom they trust; an online KP software agent (a KP reasoner) will trace KP tags (KP metadata) in web documents across web pages, combining information source and dependency, as well as trust relations, to deduce the origin and validity of tagged information.

Static KP focuses on the simplest yet strongest form of provenance. Basically, any proposition considered in Static KP has a truth value of: True, False or Unknown. As uncertainty of web information is unavoidable, Static KP temporarily uses a simple method to handle uncertainty in which “Unknown” is used to represent the status where the truth value of a proposition cannot be determined. Uncertain truth value represented with subjective probability is introduced in Uncertain KP. A Static proposition's truth value does not change over time. Dynamic truth values are considered in Dynamic KP.

The underlying concepts of Static KP are explored in the context of two case studies as follows.

Case 1: Asserted Information

Consider the proposition found on a web page that "perennial sea ice in the Arctic is melting faster than previously thought at a rate of 9 percent per decade." From a provenance perspective, there are three questions that have to be answered: 1) What is the truth value of this proposition? 2) Who asserted this proposition? 3) Should we believe the person or organization that asserted it? In this example, a further examination of the text of the web page provides the answers (www.gsfe.nasa.gov/topstory/2002/1122seaice.html): It can be believed as a true proposition, asserted by NASA, who most people believe is an authority on the subject. Questions are: how can this provenance information be represented directly without having to resort to Natural Language Processing of the page? And what is the basis for us to believe this proposition as true?

Case 2: Dependent Information

Consider the following information found in another web page: "The accelerated rate of reduction of perennial sea ice in the Arctic will lead to the extinction of polar bears within 100 years." This is actually two propositions composed of a premise, "The accelerated rate of reduction of perennial sea ice in the Arctic" and a conclusion, "the extinction of polar bears within 100 years." Just like in the previous case, three same questions need to be answered for each proposition. What makes this case more interesting is that the truth value of a proposition is dependent upon other propositions that may be found in other web pages. These types of propositions are called "dependent propositions" in KP.

There are two types of dependency occurring. First, it is common to find that a proposition in a web document is the reproduction of another proposition in another web document, e.g., quotation or rephrasing. This reproduction makes the truth value of the reproduced proposition equivalent to the truth value of the source proposition. This type of propositions is classified as “equivalent propositions”. Second, a proposition can be a conclusion derived from some premises using logical deduction. Hence, the truth value of the derived proposition depends on the truth values of its premises. This type of derived proposition is classified as "derived information".

To determine provenance, a link from a dependent proposition to the proposition it depends on is needed. This type of links or dependencies also needs to be signed and certificated together with dependent propositions.

In practice, a proposition may be derived by applying different axioms. For example, according to the demerit point system of Ontario's Ministry of Transportation, a person may get 3 points for the following reasons: Failing to yield the right-of-way; Failing to obey a stop sign, traffic light or railway crossing signal; Going the wrong way on a one-way road. Each may be a possible reason for a loss of points. Therefore, derived propositions may also be dependent upon disjunctions, conjunctions and/or negations of other propositions.

From these two cases, a number of concepts required for reasoning about provenance emerge:

- Text is divided into propositions. Once so designated, they are assumed to be indivisible.
- An asserted proposition must have a digital signature.

- If the assertion is to be believed, then the person or organization that signed the assertion must be acceptable to the user of the information.
- As propositions are reused across the web, a link between where it is used and where it came from must be maintained. These links, or dependencies, must also be signed together with propositions.
- Dependencies can be simple copies, or can be the result of a reasoning process. If the latter, then axioms used in the reasoning should also be identified and signed by an acceptable organization.

Finally, throughout the above points, the notion of acceptable signing authorities is basic to the analysis of provenance. Consequently, KP is context sensitive, where the context is defined by a set of signing authorities acceptable to the person requesting provenance.

3 Static Knowledge Provenance Ontology

What Static KP needs to answer, called informal competency questions that define requirements for the model to de developed, is identified as follows.

- Is this proposition true, false, or unknown?
- Who created this proposition?
- Is the creator of the proposition trusted?
- Does the truth of this proposition depend on any other propositions? If so, what?
- What is the digital signature verification status of this proposition?
- Which knowledge fields does this proposition belong to?
- In these fields, can the information creator be trusted?

Terminology

The taxonomy of the propositions in KP is illustrated in figure 1. KP_prop is the most general class of propositions; An Asserted_prop is an assertion that is not dependent on any other propositions; A Dependent_prop is a proposition which truth is dependent on other propositions; An Equivalent_prop is a quotation that is a copy and its truth value is the same as the proposition it depends on; A Derived_prop is a derived conclusion based on some premises; A Composite_prop could be the “and”/ “or” / “negation” of other proposition(s).

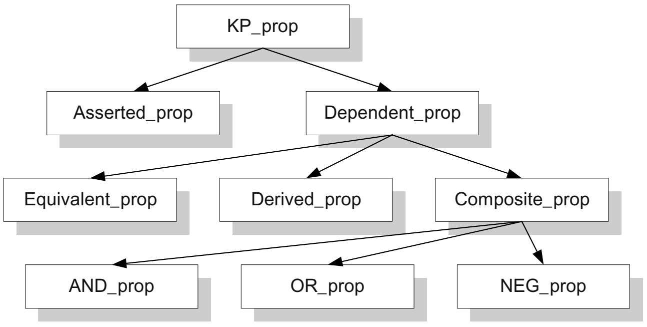


Fig. 1. Proposition Taxonomy in Knowledge Provenance

Every Asserted_prop or Derived_prop has an "*assigned truth value*" that is the truth value given by the proposition creator. And every KP_prop has a "*trusted truth value*" that is evaluated and trusted by a specific provenance requester.

A trust relation in Knowledge Provenance is defined as a of triple (*a, c, f*) where the provenance requester *a* "trusts" (directly or indirectly) information creator *c* in a topic or a specific knowledge field *f*, here, "trust" means that *a* believes any proposition created by *c* in field *f* to be true; "indirect trust" means that *a* do not directly know *c* but trusts *c* by the media of some references who trust *c*. (Note: The mathematical definition of a trust relation should be a set of triples {(*a, c, f*)}. For simplicity, a triple (*a, c, f*) is called a trust relation in this paper).

Static KP Axioms

The axioms for static KP are summarized as follows. The formal specification of these axioms in First Order Logic can be found in [Fox&Huang2003].

- A KP-prop is "trusted", if the creator or publisher of the proposition is "trusted" in a field that covers[*] one of the fields of the proposition, and the digital signature verification status is "Verified".
- For an asserted, or derived, or equivalent KP-prop that has no creator specified, the creator of the document is the default creator of the KP-prop.
- If a proposition does not have a creator, then the digital signature verification status of the KP-prop is determined by the digital signature verification status of the document.
- The default assigned truth value of a KP-prop is "True". That is, if a proposition creator does not give the truth value of a proposition, the creator implicitly declare the truth value is "True".
- The trusted truth value of an asserted-prop is the same as its assigned truth value, if the asserted-prop is trusted by the provenance requester; otherwise the trusted truth value is "Unknown".
- The trusted truth value of an equivalent-prop is the same as the trusted truth value of the proposition it depends on, if this equivalent-prop is trusted; otherwise the trusted truth value is "Unknown".
- The trusted truth value of a derived-prop is the same as its assigned truth value, if the derived-prop is trusted and the KP-prop it depends on is "True"; otherwise the trusted truth value is "Unknown". Note that it is unnecessary to include everything used to derive the truth value in the dependency.
- The trusted truth value of a negative-prop is the negation of the trusted truth value of the KP-prop it depends on, if the negative-prop is trusted by the provenance requester; otherwise the trusted truth value is "Unknown".
- The trusted truth value of an And-prop is "True" if all the KP-props it depends on are "True"; The trusted truth value of an And-prop is "False" if at least one of the KP-props it depends on is "False"; and the trusted truth value of an And-prop is "Unknown" if at least one of the KP-props it depends on is "Unknown" and none of them is "False".
- The trusted truth value of an Or-prop is "True" if at least one of the KP-props it depends on is "True"; The trusted truth value of an Or-prop is "False" if all of the

[*] The relations among different knowledge fields could be very complex, which is beyond our topic on KP. We assume that a common recognized taxonomy of knowledge fields is used.

KP-props it depends on are "False"; and the trusted truth value of an Or-prop is "Unknown" if at least one of the KP-props it depends on is "Unknown" and none of them is "True".

4 KP Annotation, Reasoning, and Implementation

In order to apply Knowledge Provenance in practice, information creators need to annotate web documents with KP metadata, information users (provenance requesters) need to define their trust relations, and KP software agent needs to be developed to conduct provenance reasoning on annotated web documents. This section illustrates how to embed Knowledge Provenance information into web documents and how it is used to determine the provenance of propositions.

In order to annotate web documents with KP metadata and define trust relations, we have defined a KP markup language in RDFS (RDF Schema, see http://www.w3.org/TR/2004/REC-rdf-schema-20040210/). The following is a piece of example containing only one proposition in a web document annotated with kp metadata. The entire annotation example can be found in [Fox&Huang2003].

```
<kp:Derived_prop rdf:id="EndangeredPolarBears"
      is_dependent_on="#MeltingArcticSeaIce"
      creator ="Andrew Derocher"
      in_field ="Polar Bears"
>
   <kp:proposition_content> The melting sea ice threatens to drive polar bears
extinct within 100 years
   </kp:proposition_content>
</kp:Derived_prop>
```

A Static KP reasoner has been implemented in Prolog. Figure 2 illustrates the provenance reasoning for the second case in section 2.

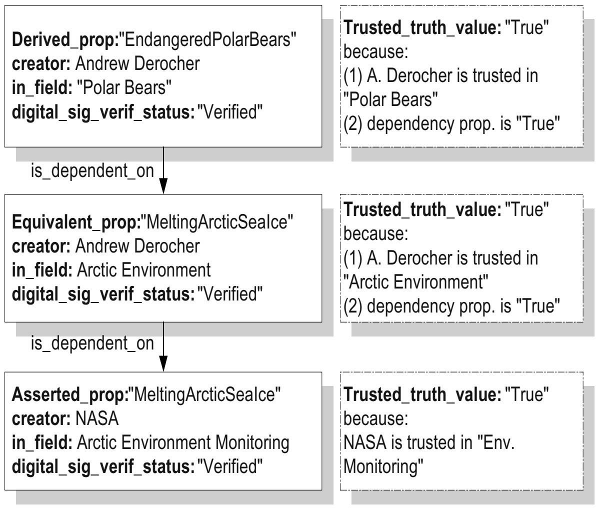

Derived_prop:"EndangeredPolarBears"
creator: Andrew Derocher
in_field: "Polar Bears"
digital_sig_verif_status: "Verified"
Trusted_truth_value: "True"
because:
(1) A. Derocher is trusted in "Polar Bears"
(2) dependency prop. is "True"
is_dependent_on
Equivalent_prop:"MeltingArcticSeaIce"
creator: Andrew Derocher
in_field: Arctic Environment
digital_sig_verif_status: "Verified"
Trusted_truth_value: "True"
because:
(1) A. Derocher is trusted in "Arctic Environment"
(2) dependency prop. is "True"
is_dependent_on
Asserted_prop:"MeltingArcticSeaIce"
creator: NASA
in_field: Arctic Environment Monitoring
digital_sig_verif_status: "Verified"
Trusted_truth_value: "True"
because:
NASA is trusted in "Env. Monitoring"

Fig. 2. Example of Provenance Reasoning

5 Summary and Future Work

Knowledge Provenance is proposed to address the problem about how to determine the validity and origin of information/knowledge on the web by means of modeling and maintaining information source and dependency as well as trust structure. Four levels of Knowledge Provenance are introduced: static, dynamic, uncertain and judgment-based. A static KP model is designed and formally specified as an ontology; a KP markup language is designed with RDFS; a KP reasoner that traces the web documents annotated with kp metadata and deduces the origin and validity of requested information is implemented in Prolog. KP model is general and could be applied to various types of wed documents including html and xml. Since each provenance requester defines his/her trust relations, KP is personalized, i.e., each requester has his/her own view of the validity of a proposition.

This static KP model has been extended to dynamic KP model [Huang& Fox2003A] that determines the validity of information in a world where the validity of a proposition and trust relations are changing over time. Furthermore, an uncertainty-oriented KP model introducing “trust degree” to represent uncertain trust relations and “degree of certainty” to represent uncertain truth value has been developed [Huang&Fox2003B].

We will continue our work towards judgment-based KP to develop a formal "social" process describing trust propagation in social networks. In addition, a web based KP reasoner will be implemented to deduce the origin and validity of requested web information by tracing web documents across the web.

This research was supported, in part, by Bell University Laboratory.

References

1. Berners-Lee, T., (1998), Semamtic Web Road Map, http://www.w3.org/DesignIssues/Semantic.html (fetched on 01 Feb. 2002).
2. Berners-Lee, T., (2003), Semantic Web Status and Direction, ISWC2003 keynote. http://www.w3.org/2003/Talks/1023-iswc-tbl/ (fetched on 23 Oct. 2003)
3. Blaze, M., Feigenbaum, J. and Lacy, J., (1996), Decentralized Trust Management, Proceedings of IEEE Conference on Security and Privacy, May, 1996.
4. Fox, M. S., and Huang, J., (2003), "Knowledge Provenance: An Approach to Modeling and Maintaining the Evolution and Validity of Knowledge", EIL Technical Report, University of Toronto. http://www.eil.utoronto.ca/km/papers/fox-kp1.pdf
5. Golbeck, J., Hendler, J., and Parsia, B., (2002), Trust Networks on the Semantic Web, Research report of University of Maryland, College Park.
6. Huang, J. and Fox, M. S., (2003A), " Dynamic Knowledge Provenance ", EIL Technical Report, University of Toronto, 2003. http://www.eil.utoronto.ca/km/papers/kp2-TR03.pdf
7. Huang, J., and Fox, M.S., (2003B), "An Uncertainty-oriented Knowledge Provenance Model", EIL Technical Report, University of Toronto.
8. Khare, R., and Rifkin, A., (1997), "Weaving and Web of Trust", World Wide Web Journal, Vol. 2, No. 3, pp. 77-112.
9. Mayorkas, A. N., (2000), http://www.usdoj.gov/usao/cac/pr/pr2000/003.htm (fetched on 01 Sept. 2002).
10. Simon, E., Madsen, P., Adams, C., (2001), An Introduction to XML Digital Signatures, http://www.xml.com/pub/a/2001/08/08/xmldsig.html (fetched on 01 Oct. 2002).

A Unified Action Language Framework

Aaron Hunter

Simon Fraser University
amhunter@cs.sfu.ca

Abstract. Action languages are simple formal languages for describing the effects of actions. Although they have been useful formalisms for reasoning about actions and planning, no precise definition has been given to specify exactly what constitutes an action language. In this paper, we introduce a logical framework that unifies existing action languages. The framework is based on a simple modal logic with one-step temporal modalities and a causal operator.

1 Introduction

Action languages are simple formal languages for describing the effects of actions. Many action languages have been defined for the representation of a wide range of action effects [1,3,4]. However, there is no precise definition that specifies exactly what constitutes an action language. In this paper, we introduce a generic formalism that unifies existing action languages.

Defining a unifying action language formalism is useful because it facilitates the proof of general results about the action language framework. Currently, the framework is actually a collection of independently defined languages. As a result, if we are interested in the expressive power of action languages, we are generally limited to comparing specific action languages with each other, or with more general formalisms (e.g. [5]). A generic underlying formalism would also facilitate the uniform extension of all action languages to represent some new phenomenon. At present, extensions in expressive power are generally defined for individual action languages, as in [2].

The main contribution of this paper is the specification of a flexible formalism that has instantiations that are provably equivalent to a wide range of popular action languages. The formalism is built by extending a modal temporal logic with a causal operator similar to that employed by McCain and Turner [7]. Essentially, each interpretation of the causal operator corresponds to a different action language.

The paper is organized as follows. In §2, we introduce generalized action languages. In §3, we give some examples to demonstrate how existing action languages can be represented in the new framework. We conclude with some discussion of future work.

2 A Unified Framework

2.1 Abstracting Structure

An action signature is a triple $\sigma = \langle \mathbf{A}, \mathbf{F}, \mathbf{V} \rangle$ where $\mathbf{A}$ is the set of action symbols, $\mathbf{F}$ is the set of fluent symbols and $\mathbf{V}$ is the set of fluent values. For simplicity, we restrict attention to action signatures where $\mathbf{V}$ is the set $\{t, f\}$ of propositional truth values.

We assume the reader is familiar with the action languages $\mathcal{A}$ and $\mathcal{C}$ [4]. These are typical examples, defined by specifying a class of *propositions*, and associating a transition system with every set of propositions. A set of propositions is called an *action description*. We remark that, identical propositions need not have the same meaning in different action languages. Nevertheless, keywords such as **if** and **after** are frequently used for the same purpose, as in the following examples:

$$(\mathcal{A}\text{ proposition})\quad A\ \textbf{causes}\ F\ \textbf{after}\ G \tag{1}$$

$$(\mathcal{C}\text{ proposition})\quad \textbf{caused}\ F\ \textbf{if}\ H\ \textbf{after}\ A \wedge G \tag{2}$$

We can abstract a general pattern from these examples. We introduce unary modalities pre and succ along with a causal operator $\Leftarrow$. Informally, a sentence of the form $\mathsf{pre}(\psi)$ is true just in case ψ holds in the previous state. Similarly, succ denotes a successor operator and $\Leftarrow$ denotes a relation of causal explanation. Using these operators, we can give concise translations of propositions (1) and (2):

$$\mathsf{succ}(F) \Leftarrow A \wedge G \tag{1}$$

$$\mathsf{succ}(F) \Leftarrow A \wedge G \wedge \mathsf{succ}(H). \tag{2}$$

By separating temporal information from causal information, we get a simple language that identifies the common temporal structure of action languages, while isolating the causal operator that may vary.

Formally, let AL denote Propositional Temporal Logic restricted to include only two temporal modalities pre and succ. The syntax and semantics for this logic can be found in [6]; we give only a couple of important definitions. A *model* is a pair $\langle \delta, I\rangle$ where δ is an ordered set of *states* and I maps each $s \in \delta$ to an interpretation $I(s)$ of σ. Formulas of AL are evaluated at triples $\langle \delta, I, j\rangle$ where $\langle \delta, I\rangle$ is a model and j is a natural number. A formula is true in $\langle \delta, I, j\rangle$ if it is true in the j^{th} state of δ.

2.2 Generalized Action Languages

We say that an AL formula ϕ is *simple* if it has the form $\mathsf{pre}^n P$ or $\mathsf{succ}^n P$, for some non-modal formula P. Intuitively, a simple formula makes a propositional assertion about a single state. We give a general syntactic definition of a proposition.

Definition 1. *A proposition of AL is an expression of the form $\phi \Leftarrow \psi$, where ϕ and ψ are conjunctions of simple AL formulas. $PROP(\sigma)$ denotes the set of all propositions over the action signature σ.*

The formula on the left side of a proposition is called the *head* and the formula on the right is called the *body*.

We would like to be able to say when a proposition is consistent with a triple $\langle \delta, I, j\rangle$. For example, a proposition of the form

$$\mathsf{succ}(\mathit{InAir}) \Leftarrow \mathit{Jump}$$

is consistent with $\langle (s_0, s_1), I, 0\rangle$ provided that $I(s_0)(\mathit{Jump}) = t$ and $I(s_1)(\mathit{InAir}) = t$. The collection of all models that are consistent with a proposition will give a picture of the causal relationship expressed.

Definition 2. *Let INT denote the set of all triples $\langle \delta, I, j\rangle$ over σ. A description interpretation is a pair $\langle \mathsf{P}, \rhd\rangle$ where*

- $\mathsf{P} \subseteq PROP(\sigma)$
- $\rhd \subseteq INT \times 2^{2^{\mathsf{P}}}$

We think of $\rhd$ as a satisfaction relation giving necessary and sufficient conditions for an action description to be satisfied by $\langle \delta, I, j\rangle$. Hence, a description interpretation specifies a class of admissible propositions and it associates a set of models with each action description.

Generalized action languages are description interpretations that satisfy a natural requirement called *local consistency*. Informally, a description interpretation is locally consistent if all models that are equivalent on a suitable interval satisfy the same action descriptions. The formal details of the definition are not important for the present paper. We simply remark without discussion that the local consistency requirement does not rule out any plausible account of causal explanation.

Definition 3. *A generalized action language is a description interpretation $\mathcal{G} = \langle \mathsf{P}, \rhd\rangle$ in which $\rhd$ is locally consistent. An action description in $\mathcal{G}$ is a subset of* P.

Note that we have not restricted the interpretation of $\rhd$ to address any of the common problems in reasoning about action. For example, we have not considered the persistence of fluent values. Instead, the treatment of such problems must be addressed when a specific generalized action language is introduced. In this manner, we do not commit to any specific treatment of, for example, the frame problem.

3 Encoding Existing Formalisms

3.1 A Family of Causal Operators

In this section, we give a family of relations $\rhd_i$ based on McCain and Turner's notion of causal explanation. If AD is an action description, define the *reduct* $AD^{\langle \delta, I, j\rangle}$ to be the set of heads of propositions in AD whose bodies are satisfied by $\langle \delta, I, j\rangle$. The succ^n reduct of AD is the propositional formula $AD^{\langle \delta, I, j\rangle}[\mathsf{succ}^n]$, obtained from the conjunct of $AD^{\langle \delta, I, j\rangle}$ involving n succ operators. Define $\rhd_i$ as follows.

$$\langle \delta, I, j\rangle \rhd_i AD \iff I(s_{j+i}) \text{ is the only model of} AD^{\langle \delta, I, j\rangle}[\mathsf{succ}^i]$$

Hence $\langle \delta, I, j\rangle \rhd_i AD$ holds just in case the atomic propositions true in i steps are precisely those that are explicitly caused to be true by AD. Therefore, fluent values are not persistent by default; if we want to specify that some fluent is inertial, we must explicitly do so in the action description.

We assume the reader is familiar with causal theories [7]. We demonstrate that the causal theories formalism can be represented by a generalized action language. Let $\sigma = \langle \emptyset, \mathbf{F}, \{t, f\}\rangle$. The generalized action language $\mathcal{GCT}$ is the pair $\langle \mathsf{P}, \rhd_0\rangle$ where P is the set of σ-propositions with non-modal heads and bodies.

The following result follows directly from the definition of the succ^0 reduct of AD.

Theorem 1. *Let AD be a causal theory over the vocabulary* $\mathbf{F}$*, let* $\langle \delta, I \rangle$ *be a model, and let j be a natural number. Then $I(s_j)$ is a causally explained interpretation of AD if and only if* $\langle \delta, I, j \rangle \rhd_0 AD$.

Hence, causal theories can be represented by a generalized action language with satisfaction relation $\rhd_0$. In the next section, we will see that the representation of $\mathcal{C}$ relies on $\rhd_1$. Hence, the generalized action language representation explicitly illustrates the formal relationship between the notions of causal explanation underlying each framework.

3.2 Representing $\mathcal{C}$

Let AD be a $\mathcal{C}$ action description. Define AD' through the following translation.

$$\textbf{caused } F \textbf{ if } H \textbf{ after } U \quad \mapsto \quad \mathsf{succ}(F) \Leftarrow U \wedge \mathsf{succ}(H)$$

$$\textbf{caused } F \textbf{ if } G \quad \mapsto \quad \begin{cases} F \Leftarrow G \\ \mathsf{succ}(F) \Leftarrow \mathsf{succ}(G) \end{cases}$$

Let P be the set of propositions that can be obtained in the translation $AD \mapsto AD'$. Define $\rhd$ such that, for any $AD \subset \mathsf{P}$, we have $\langle \delta, I, j \rangle \rhd AD$ if and only if both

– $I(s_j) \models G \rightarrow F$ for each proposition $F \Leftarrow G \in AD$
– $\langle \delta, I, j \rangle \rhd_1 AD$.

The generalized action language $\mathcal{GC}$ is the pair $\langle \mathsf{P}, \rhd \rangle$.

The following theorem states the formal expressive equivalence of $\mathcal{C}$ and $\mathcal{GC}$.

Theorem 2. *The AD be a $\mathcal{C}$ action description, and let T_{AD} be the associated transition system. There is an edge (s_0, a, s_1) in T_{AD} if and only if $\langle (s_0, s_1), I, 0 \rangle \rhd AD'$, where I and T_{AD} assign the same interpretations to s_0 and s_1.*

Proof. Let (s_0, a, s_1) be an edge in T_{AD}. Hence, the interpretation that T_{AD} associates with s_1 is the only interpretation that satisfies all static rules in AD as well as the heads of all applicable dynamic rules in AD. So $I(s_1)$ is the only interpretation that satisfies all propositions of the form $\mathsf{succ}(F) \Leftarrow \mathsf{succ}(G)$ along with the heads of all applicable propositions of the form $\mathsf{succ}(F) \Leftarrow U \wedge \mathsf{succ}(H)$. Hence $\langle \delta, I, j \rangle \rhd_1 AD'$. Moreover, since s_0 is a state in a $\mathcal{C}$ transition system, it follows that $I(s_0) \models G \rightarrow F$ for every proposition of the form $F \Leftarrow G$ in AD'. Therefore, $\langle \delta, I, j \rangle \rhd AD'$.

The converse is similar.

Similar translations can be used to represent $\mathcal{A}$ and $\mathcal{B}$ as generalized action languages.

3.3 Non-standard Action Languages

In this section, we have focused on the representation of $\mathcal{C}$ because it has been one of the most popular action languages. However, generalized action languages can also represent non-standard action effects that are normally treated in specialized action languages. For example, non-Markovian action languages like the one in [3] can be represented by using nested pre modalities in proposition bodies. Similarly, *action triggers* of the kind described in [1] can be represented by including conjuncts of the form $\mathsf{succ}^n A$ in proposition heads.

4 Conclusion

The generalized action language framework defined in this paper represents a first step towards abstracting a single underlying action language formalism. The framework illustrates that it is possible to abstract a flexible syntax that intuitively captures the kind of statements that are possible in an action language. In the case of standard action languages like $\mathcal{C}$, the corresponding generalized action languages are just superficial variations on the syntactic form of propositions. However, by writing propositions in a common syntactic form, some expressive differences between languages become clear.

One important feature of the new language is that it separates temporal information from causal information. This separation allows us to use a well-known modal logic to represent temporal information, and it makes it salient that the distinction between many action languages can be captured by interpreting a single causal operator.

Currently, the only restriction on the interpretation of the causal operator is that it must be locally consistent. In the future, we would like to further restrict the admissible interpretations to capture a more reasonable notion of causal explanation. Placing intuitive restrictions on the semantics helps to clarify exactly what should be representable in an action language. Moreover, such restrictions also facilitate the proof of universal properties that hold for all generalized action languages.

In order to demonstrate the utility of generalized action languages, we are currently working on an epistemic extension of the framework. By extending the general framework, we can uniformly define epistemic extensions of the action languages in the $\mathcal{C}$ family, simply by looking at the corresponding instantiations. This will extend the range of application of these action languages to a new class of problems.

References

1. C. Baral and N. Tran. Representation and reasoning about evolution of the world in the context of reasoning about actions. In *Proceedings of The Workshop onNonmonotonic Reasoning, Action, and Change (NRAC'03)*, pages 31–36, August 2003.
2. T. Eiter, W. Faber, N. Leone, G. Pfeifer, and A. Polleres. Answer set planning under action costs. In G. Ianni and S. Flesca, editors, *Proceedings of the 8th European Conference on Artificial Intelligence (JELIA)*, volume 2424 of *Lecture Notes in Artificial Intelligence*, pages 541–544. Springer-Verlag, September 2002.
3. M. Gelfond, C. Baral, and G. Gonzalez. Alan: An action language for non-markovian domains. In *IJCAI-03 Workshop on Nonmonotonic Reasoning, Action and Change (NRAC03)*, 2003.
4. M. Gelfond and V. Lifschitz. Action languages. *Linköping Electronic Articles in Computer and Information Science*, 3(16):1–16, 1998.
5. E. Giunchiglia and V. Lifschitz. Action languages, temporal action logics and the situation calculus. *Linköping Electronic Articles in Computer and Information Science*, 4(40):1–19, 1999.
6. O. Lichtenstein and A. Pnueli. Propositional temporal logics: Decidability and completeness. *Logic Journal of the IGPL*, 8(1):55–85, 2000.
7. N. McCain and H. Turner. Causal theories of action and change. In *Proceedings of AAAI-97*, pages 460–465, 1997.

An Automatic Evaluation Framework for Improving a Configurable Text Summarizer

Loïs Rigouste[†], Stan Szpakowicz[⋆], Nathalie Japkowicz[⋆], and Terry Copeck[⋆]

[⋆] SITE, University of Ottawa, Ottawa, Canada
[†] École Nationale Supérieure des Télécommunications, Paris, France

Abstract. CALLISTO is a text summarization system that depends on machine learning techniques and is therefore sensitive to pre-established biases that may not be wholly appropriate. We set out to test whether other biases, modifying the space that CALLISTO explores, lead to improvements in the overall quality of the summaries produced. We present an automatic evaluation framework that relies on a summary quality measure proposed by Lin and Hovy. It appears to be the first evaluation of a text summarization system conducted automatically on a large corpus of news stories. We show the practicality of our methodology on a few experiments with the Machine Learning module of CALLISTO. We conclude that this framework gives reliable hints on the adequacy of a bias and could be useful in developing automatic text summarization systems that work with Machine Learning techniques.

1 Introduction

Automatic text summarizers either generate an abstract from some semantic representation of the document, or extract relevant document fragments, usually sentences. Our system, CALLISTO [Copeck *et al.*, 2002], is in the latter category. To find the sentences for a summary, it dynamically selects a text segmenter and a key phrase extractor, choosing among several versions of each module. We use Machine Learning (ML) to find which modules tend to generate good summaries for what types of documents. Training generalizes a large set of raw summaries into rules that pick the best configuration given a document's properties. The well-known ML system C5.0 helps determine the segmenter and key phrase extractor best suited to a given text, but it was an arbitrary decision to use C5.0 and to choose other system settings.

CALLISTO generates such volume of intermediate data[1] that no human judge can claim with confidence that any design decisions are better than others, let alone optimal. We need a systematic way of measuring the effect of various system settings on the quality of the summaries produced. We describe automated evaluation that attempts to meet this need by allowing us to compare and choose between different settings of system parameters. While our framework has been tested on CALLISTO, it could fit other ML-based summarizers.

[1] Our training data, obtained courtesy of the Document Understanding Conferences, contain over 1100 texts; the number of summaries generated exceeds 300,000.

We consider evaluation of automatic text summarizers and explain how this has affected our design of an evaluation framework that automatically measures the performance of a given version of CALLISTO. The results of a few experiments support our expectation that such a framework helps CALLISTO improve the overall quality of summaries.

2 Evaluation in Automatic Summarization

It is difficult to evaluate summary quality in general. One may test a summarizer on any task in which a summary replaces the whole document. Human judges are needed in such *extrinsic* methods [Mani, 2001] that are too costly for us (we cannot afford evaluating thousands of summaries). *Intrinsic* methods are intended to give an absolute measure of either the *quality*, how well a summary is written, or the *informativeness*, how much knowledge it contains. We extract *correct* sentences from the document, so our priority is informativeness.

Our training data come with model summaries, so comparison with models may measure summary quality well. Human judges must compare generated summaries with models comprehensively [DUC, 2001 and 2002], but automatic methods based on content words do compute simple measures of closeness between a summary and a model. Mani [ibid.] suggests that such methods may succeed in a case like ours. We work with news reports; model summaries tend to borrow material from the source. A good extract may have a lot in common with the model abstract if it contains the most relevant sentences.

A measure is trustworthy if it can be shown to correlate well with human judgment. That is the topic of a study [Lin and Hovy, 2002] of a family of measures based on content words. Section 3 presents their method and explains how we modified it for our evaluation framework.

3 The Proposed Evaluation Framework

Lin and Hovy's Measure. Lin and Hovy's work was inspired by IBM's method of evaluating output of Machine Translation [Papieni *et al.*, 2001], BLEU. It compares a text with model translations by matching N-grams (length 1-4). Lin and Hovy adapt this measure to text summarization, and show that it agrees well with human judgment[2]. They used only recall (how many N-grams from the model abstract are in the summary?) instead of BLEU's modified weighted precision that considers multiple reference translations.

Lin and Hovy considered several N-gram matching methods: with or without stemming, with several weights for various N-grams (chiefly unigrams and bigrams). Then they proposed a ranking for the systems that participated in DUC 2001, and compared it with the official ranking established by human judges. *All* the methods considered correlate with human evaluation at least as well

[2] A continuation of that work [Lin and Hovy, 2003] has led to the "ROUGE" metric [Lin, 2003], adopted as the evaluation criterion in the DUC 2004 exercise.

as two people agree. Mechanical assessment will not achieve perfect agreement with human judgment if human evaluation is imperfect; and there cannot be *one* accurate human rating.

Our framework implements a method that agrees more closely with the official DUC ranking. Weights of 1/3 are applied to unigram and 2/3 to bigram scores produced by Porter's stemmer after removing stop words from the text. This produces reference lists of stemmed unigrams (length N) and bigrams (length $N-1$) from the model summary. We similarly stem the summary under evaluation and count its unigrams and bigrams appearing in the reference lists. This number is the overlap. Our recall measure is:

$$Recall = \frac{1}{3}\frac{Overlap_{Unigrams}}{N} + \frac{2}{3}\frac{Overlap_{Bigrams}}{N-1}$$

F-Score. A tested summary should agree with the model not only in content but also in length, an aspect measured by *Precision* (the same formula as Recall, replacing N by the length of the summary to test). Combining informativeness and brevity is possible through F-Score, a measure used in Information Retrieval:

$$\text{F-Score} = 2 * \text{Recall} * \text{Precision}/(\text{Recall} + \text{Precision})$$

In [Papieni *et al.*, 2001], using F-Score as such was not possible because of multiple references. We do not have this problem, so we can consider F-Score rather than Recall. We only need to prove the correlation with human judgment. As discussed in [Lin and Hovy, 2002], the important point is to preserve ranking. In our dataset of 334210 summaries it is easy to see if the measures correlate. We computed the correlation of Recall and F-Score with the Spearman rank-order coefficient and obtained $\rho = 86.94\%$. Recall correlates well with human judgment at more than 98% [Lin and Hovy, 2002] and the correlation between Recall and F-Score is satisfactory. We can assume that F-Score correlates well with human judgment. The transitivity rule we use here is only acceptable because the correlation scores are very high. It is, strictly speaking, only true when the Spearman coefficient is 100%, but, as both scores are high, we can assume good correlation between F-Score and human judgment.

Baseline and Statistical Significance. [Brandow *et al.*, 1995] show that a very efficient baseline for automatic summarization of news reports is to extract the first few sentences (approximately 1-2 paragraphs). This can be explained by the journalistic style: in a news report, one expects the answer to the typical questions (who, when, where, what, why) in the first lines. This can be deemed an acceptable summary in many cases. Therefore, it is important to include such a challenging baseline in every evaluation study. Note that it is highly genre-specific. In other kinds of texts, it could be much less interesting.

We have extracted the first sentences of each text in the corpus and run Lin and Hovy's evaluation method on them. As expected, and as we will see in the next section, this baseline is extremely challenging. To establish the statistical significance thresholds of our results, we used two-way ANOVA [Hull, 1993].

4 An Experiment on CALLISTO Using the Framework

Experiment. The learning component is a crucial part of CALLISTO since it is responsible for predicting the right configuration to apply to each text. Therefore, the main idea underlying the system – that the setting to choose depends on the input text – is valuable only if we can efficiently take advantage of knowledge in the training data to predict choices for further texts.

C5.0 was chosen as a fast, efficient state-of-the-art program. It does happen that other learners work better for a certain application, and it could be the case for us. Out of the wide range of Machine Learning algorithms we considered, very few were applicable to our dataset because of the large number of examples. We present here the only one that brought significant improvement in comparison to C5.0: Naive Bayes.

Without explaining in detail how CALLISTO works, we have to mention a discretization process that precedes learning and is done with K-means. We do not know which number of classes is best so we have generally tested every value between 2 and 20. We report here only the most interesting results. See [Rigouste *et al.*, 2004] for more details and other experiments.

Results. The experiments were conducted with tenfold cross-validation. We present in the following table the averaged F-Score of each version of the system. The statistical significance is .01 and the baseline score is .246.

Number of classes	2	3	5	8	10	15	20
F-Score with Naive Bayes	.231	.234	.230	.233	.229	.220	.219
F-Score with C5.0	.222	.225	.224	.222	.216	.218	.219

Discussion. The baseline beats every version of the system we test here. It must be due to the way journalists write, as explained earlier, and may not work well for other genres. Naive Bayes significantly outperforms C5.0 for almost every number of classes. F-Score is lower for large numbers of classes, probably due to overfitting. The optimal number of classes seems to be 3 in this experiment.

It is unclear why Naive Bayes gives in the end better results than C5.0 but it is interesting to notice that a quick study of the error rates of the algorithms would bring an opposite conclusion. When we look at the accuracy of the classifiers on our dataset in 10-fold cross validation, C5.0 seems better than Naive Bayes. However, we are only interested in a small subset of the predictions, corresponding to the best class and made with high confidence. It is possible that on those, Naive Bayes outperforms C5.0[3].

This analysis demonstrates the usefulness of our framework. Based only on the global error rates, we would have preferred C5.0, without necessarily having the idea to investigate further and examine the error rate on the configurations

[3] This is in part confirmed by experiments in [Rigouste, 2003]. For Naive Bayes, the error rate is better on the predictions actually used; the effect is opposite for C5.0.

selected. With the framework we judge configurations of the system on the overall qualities of the summaries, that is, on the final output of the system, and not on an intermediate measure such as the error rate of the learning component.

5 Conclusions and Future Work

We have presented an evaluation framework for automatic summarization systems. The framework, based on a measure proposed by Lin and Hovy, is fully automatic and can be applied to large corpora, which gives more significance to the results. It comes with a very demanding baseline: lead text extraction. The framework helped us make decisions based on what we are really interested in (producing better summaries) and not on intermediate measures that have an unclear effect on the final performance of the system.

For the framework to be deemed trustworthy, a reliable experiment with human judges would be necessary. Another promising direction of future work is to keep using the framework to test other design choices, such as the features that characterize document well.

Acknowledgements. NIST, the organizer of the Document Understanding Conferences, is instrumental in summarization evaluation. NSERC partially supported the first author.

References

[Brandow *et al.*, 1995] Brandow, R., Mitze, K., Rau., L. Automatic condensation of electronic publications by sentence selection. *Information Processing and Management*, 31(5):675-685.

[Copeck *et al.*, 2002] Copeck, T., Japkowicz, N., Szpakowicz, S. Text Summarization as Controlled Search. *Canadian AI'2002*. 268-280.

[DUC, 2001 and 2002] Document Understanding Conferences, National Institute of Standards and Technology. http://duc.nist.gov/

[Hull, 1993] Hull, D. Using statistical testing in the evaluation of retrieval experiments. *SIGIR '93*, 329-338.

[Lin, 2003] Lin, C.-Y. ROUGE, Recall-Oriented Understudy for Gisting Evaluation. http://www.isi.edu/˜cyl/ROUGE/.

[Lin and Hovy, 2002] Lin, C.-Y., and Hovy, E.H. Manual and Automatic Evaluations of Summaries. *Workshop on Automatic Summarization*, ACL-02.

[Lin and Hovy, 2003] Lin, C.-Y., and Hovy, E.H. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. *HLT-NAACL 2003*. 150-157.

[Mani, 2001] Mani, I. *Automatic Summarization*. John Benjamins.

[Papieni *et al.*, 2001] Papieni, K., Rouckos, S., Ward, T., and Zhu. W.-J. BLEU: a Method for Automatic Evaluation of Machine Translation. IBM Research Report.

[Rigouste, 2003] Rigouste, L., Evolution of a Text Summarizer in an Automatic Evaluation Framework. Master's thesis. http://www.site.uottawa.ca/˜rigouste/thesis.ps

[Rigouste *et al.*, 2004] Rigouste, L., Szpakowicz, S., Japkowicz, N., Copeck, T., An Automatic Evaluation Framework for Improving a Configurable Text Summarizer. TR-2004-01, SITE, University of Ottawa. http://www.site.uottawa.ca/˜szpak/recent_papers/TR-2004-01.pdf

An Algorithm for Anaphora Resolution in Aviation Safety Reports

Katia Dilkina and Fred Popowich

Simon Fraser University, Burnaby, BC, CANADA, V5A 1S6
`http://www.sfu.ca/~knd/`
`http://www.cs.sfu.ca/~popowich/`

Abstract. While the general problem of determining the referent of a pronoun like *it*, *he* or *she* is quite difficult, by constraining the domain of discourse, and by constraining the task, it is possible to achieve accurate resolution of the antecedents of pronouns in certain contexts. In this paper, we constrain the domain of discourse by considering only aviation safety reports, and consider the task of pronominal resolution in the context of entity extraction in these aviation texts. In doing so, we are able to provide an algorithm that is better suited to the task than general purpose anaphora resolution algorithms, and which should be applicable to other domains by incorporating similar constraints.

Keywords: Entity extraction, pronouns, reference resolution

1 Introduction

Anaphora resolution is a hard problem; a great deal of linguistic and non-linguistic knowledge must be brought to bear to determine the 'meaning' of pronouns (like *it*, *they* and *them*) when they occur in texts. As is often done in natural language understanding tasks, the problem can be simplified by constraining the domain of discourse and by constraining the context in which natural language is used. We constrain the domain of discourse by dealing only with aviation safety text, and we constrain the context by considering only the genre of text found within safety reports, and by considering only the task of entity extraction.

Anaphora can be defined as the process of using pronouns or similar descriptors to refer to entities or expressions found elsewhere in a text, or in the domain of discourse. So, a pronoun, like *she* is an example of an anaphor, while a noun phrase like *The pilot* is an example of a possible antecedent for the anaphor in a sentence like *The pilot thought she might have to abort the landing.* Anaphora resolution can then be defined as the process of finding an appropriate antecedent for each anaphor in the text.

In looking at the techniques used for anaphora resolution, one finds that they differ depending on: 1) the types of anaphors being processed, 2) the degree to which they use syntactic constraints, and 3) their employment of focusing and centering theory techniques. Here, we will restrict our investigation to third person pronouns, specifically: *he, his, him, she, her, it, its, they, their* and *them.*

2 Aviation Safety Reports

Our ASRS corpus [1] consists of approximately 80,000 reports of incidents involving aircraft. In our study, we manually analyzed 200 of these documents, randomly selected. The average length of a report is 280 words and 16 sentences. The domain is highly specific with a lot of terminology and abbreviations. As noted in [8], about 10% of all the words contained in ASRS reports are abbreviations. For our collection of 200 documents, on average, there are 19 first and third person pronouns in each document, approximately 74% of which are first person pronouns, 22% are third person anaphoric pronouns, and 4% are pleonastic third person pronouns.

In addition to the characteristics outlined above, the ASRS corpus presents the following issues not characteristic for corpora used with anaphora resolution algorithms so far:

First of all, all text is typed in uppercase letters and there is a large number of non-standard abbreviations. Therefore, additional processing is required, with the help of a database of abbreviations and proper nouns, in order to determine the correct POS tags for all lexical items.

Secondly, there is a considerable amount of grammatical and spelling mistakes, which not only hinder basic stages of text processing such as POS tagging and parsing, but also often makes it difficult to resolve seemingly straight-forward co-reference. The seriousness of the mistakes vary from simple misspelling or faulty placement of a period, to gross grammatical errors, which make the text incomprehensible even to an intelligent human reader.

Thirdly, a phenomenon widely observed in the corpus is gender and/or number disagreement between anaphors and antecedents. Often an entity is referred to once by *he* and other times by *they*, or by *he* and *it*, etc., as seen in (1). This peculiarity of the corpus has at least one immediate consequence with respect to the design of an anaphora resolution algorithm, i.e. the algorithm could not include a person/gender/number agreement component, which is present practically in all previous coreference resolution algorithms as an effective filter for antecedent candidates for each anaphor (cf. [4], [5], [6]).

AT TIMES **ATC (= air traffic control)** *WILL GIVE AN INTERSECTION XING ALT, AND WHEN YOU ADVISE* **THEM** *THAT IT WOULD BE IMPOSSIBLE TO MAKE THE ALT* **THEY** *SAY "DO YOUR BEST." AND THE THING I DON'T UNDERSTAND IS THAT THERE IS A FIX 14 MILES WEST OF MARTINSBURG VOR THAT IS ON THE CHARTS AND WAS ALSO IN THE COMPUTER. WHY DIDN'T* **HE** *USE THAT?* (1)

Yet another difficulty arises due to the colloquial arbitrary style of this genre of text. Often it is either ambiguous or even impossible, even in manual anaphora resolution, to determine the correct antecedent of an anaphor. This happens because of ambiguity or vagueness of expression, or simply because of incoherent or incomplete thought expressed by a sentence.

In our study, apart from analyzing two hundred aviation safety reports with respect to their style, number and kinds of anaphors, and distance of the antecedent, we also looked in great detail at the nature of the antecedents. Despite their unstructured colloquial style, the ASRS data represent a very constrained domain of knowledge, and this is reflected in the use of anaphors. In particular, we found that 65% of all third person pronouns are used to refer to only four types of entities; namely, aircraft, aircraft crew, air-traffic control, and the person reporting the event (reporter).

3 An Approach to Anaphora Resolution

As we have seen, the ASRS corpus differs greatly from technical manuals corpora, which have been extensively used with previous anaphora resolution algorithms (cf. [5], [6]), and which are characterized by grammaticality and highly specialized language both in terms of lexicon and syntactic structure. The aviation safety reports are also different from journal and newspaper articles, which is the second type of corpora widely used with anaphora resolution algorithms (cf. [7], [9], [4]). Notably, journal and news articles include a range of syntactic structures as well as some amount of quotations, and so they are similar to the ASRS documents. However, they have a considerably larger knowledge domain, they do not use a great amount of abbreviations and jargon, and also they generally obey the grammatical rules of English. So, with respect to lexicon and writing style, the ASRS corpus is somewhere between technical manuals and journal and news articles. This observation leads to the conclusion that a new approach to anaphora resolution is needed if we are to be able to successfully computationally analyze aviation safety reports.

The algorithm consists of four steps, as shown below.

1. when an anaphoric pronoun a_i is identified in sentence s_k, a list of candidate antecedents is created, $\{c_1, c_2, ..., c_n\}$, which includes all named entities in s_k and s_{k-1}.
2. apply assignment function f to determines the salience value for each c_j ,
3. select c_{max} having largest value for $f(c_j)$
4. if $f(c_{max}) \geq t$, where t is the threshold salience value,
   a) then return c_{max} as the antecedent of a_i
   b) else the named entities from s_{k-2} are processed. If none of them yields a candidate value above or at the minimum, entities from s_{k-3} are processed, and so on.

Our approach differs from previous anaphora resolution systems in a number of ways. As discussed in the previous section, there is no matching for gender and number agreement between anaphors and antecedent candidates. Also, there is no morphological or deep syntactic processing – we only need a partial parser to identify the noun phrases (and in particular, we are interested in the head noun). In contrast with previous algorithms, our approach will have no limit on the possible distance between an anaphor and its antecedent. Our approach also requires a minimum threshold salience value for a candidate antecedent to be determined as an actual antecedent (our initial empirical study of the data suggests setting this number at 10).

Our pilot study shows that entities appearing as first NPs or last NPs in a sentence are preferred as antecedents. Thus, candidate antecedents at those positions will be given extra salience weights. For declarative sentences without a subject (i.e. declarative sentences which do not begin with an NP), the subject (i.e. the first NP) of the previous sentence is to be automatically assumed as the subject of the current sentence. At this preliminary stage, we are not interested in identifying full coreference chains, but only in identifying the antecedent of each anaphor. However, as we shall see below, the named entity will increase its weight by 10 as candidate antecedent for a_2 after having been determined as the correct antecedent of a_1. This reflects the fact that in ASRS documents we often see a string of sentences with the majority of pronouns referring to one and the same entity, which is named only in the beginning of the string.

Our processing will depend heavily on genre-specific and domain-specific knowledge. The assignment of salience values to candidate antecedents will depend on four factors:

1. **position in the text** — all candidate antecedents which are *in the same sentence* as the anaphor are awarded a weight of 5; all candidate antecedents which are *in the preceding sentence* are awarded a weight of 2; all other candidate antecedents are awarded a weight of 0 for this factor
2. **position in the sentence** — all candidate antecedents which are *the first noun phrase* in the sentence in which they occur are awarded a weight of 5; all candidate antecedents which are *the last noun phrase* in the sentence are awarded a weight of 2; all other candidate antecedents are awarded a weight of 0 for this factor
3. **status of the antecedent** — the candidate antecedent which has already been identified as *the actual antecedent of the previous anaphor* is awarded a weight of 10; all other candidate antecedents are given a weight of 0 for this factor
4. **available semantic information** about the particular named entity which is considered as candidate — special preference will be given to named entities belonging to the four *antecedent categories* mentioned earlier (i.e. *AIRCRAFT, REPORTER, CREW*, or *CONTROL*). The amount of salience weight given to a candidate will reflect both *the frequency of occurrence of the exact head noun* (e.g. *RPTR, WDB, CAPT*, etc.), and *the frequency of occurrence of the antecedent category.* We will assume that the frequency of occurrence of the exact head noun ($Freq_n$) is a more important factor than the frequency of occurrence of the category ($Freq_c$). Thus, the salience weight for this factor will be computed using the following formula: $round(\frac{2.Freq_n+Freq_c}{10}, 0)$.

Let us now consider how our algorithm for anaphoric pronoun resolution would be applied to the ASRS example in (1), which contains three anaphoric pronouns. We have simplified the following discussion to only include potential antecedents with scores of 2 or higher. According to our algorithm, we first calculate the set of candidate antecedents for the anaphor. So, for *THEM* in (1) we could have the candidate set $\{$*THE INFORMATION, THE COMPUTER, THE RESTRICTION, ATC, AN INTERSECTION XING ALT* $\}$. Then, the assignment function will give each candidate the following corresponding salience weights with respect to the four salience factors we defined: $\{2+0+0+0=2, 2+0+0+0=2, 2+2+0+0=4, 5+5+0+3=13, 5+0+0+0=5\}$. Only one of the values is above 10, therefore *ATC* is chosen as the actual antecedent of the anaphor *THEM*, which according to our analysis is correct. Similarly, the candidate set for *THEY* would be $\{$*ATC, AN INTERSECTION XING ALT, THE ALT* $\}$, with weights $\{5+5+10+3=23, 5+0+0+0=5, 5+2+0+0=7\}$, resulting in *ATC* also being chosen as the antecedent for *THEY*. Finally, the antecedent set for *HE* would be $\{$*ATC, AN INTERSECTION XING ALT, THE ALT, YOUR BEST, THE THING, A FIX 14 MILES WEST OF MARTINSBURG VOR, THE CHARTS, THE COMPUTER* $\}$ with weights $\{0+5+10+3=18, 0+0+0+0=0, 0+2+0+0=2, 2+0+0+0=2, 2+0+0+0=2, 2+0+0+0=2, 2+0+0+0=2, 2+2+0+0=4\}$, again resulting in *ATC* being the antecedent.

4 Analysis and Conclusion

From our ASRS corpus, we chose 10 sentences containing anaphors that were representative of the kind of constructions found in the ASRS corpus. These sentences were then tagged with a part-of-speech tagger based on the Church algorithm [3], and then parsed using the CASS partial parser [2]. A manual analysis of our algorithm, when applied to this very small sample set, showed that it could correctly identify the antecedents for 9 out of 12 anaphors. Based on this analysis, our future research will consist of an implementation of the anaphora resolution algorithm, to be followed by a detailed evaluation of the algorithm, along with a comparison with other algorithms. Our detailed evaluation will also be done in the context of examining the effect of the accuracy of the tagging, and of the partial parsing processes.

Aviation safety reports contain a high degree of domain specific terminology, including abbreviations and acronyms, and as such it is not surprising that additional factors must be taken into consideration when processing instances of anaphora and extracting entities. We have suggested an approach where semantic information about the domain of application is one of the four factors determining the referent of an anaphoric pronoun. This semantic information, together with syntactic information concerning the position of the pronoun and its possible antecedent, and together with the significance of the anaphor (whether it has been previously used as an antecedent), are used in determining salience values for possible antecedents for a pronoun. What is interesting is that some traditional information, like lexical agreement and an explicit window based on the number of sentences separating an anaphor and its antecedent, do not play an explicit role in determining the anaphor/antecedent relationship. The result is an algorithm that is able to find long distance antecedents, and find antecedents even when the anaphor and antecedent have conflicting agreement values.

Another avenue for further exploration is the automatic construction of the semantic model used by the anaphora resolution algorithm, particularly the tuning of weights for the different semantic classes. Research into this task would also allow the anaphora resolution and entity extraction techniques to be applied to other domains containing a restricted set of entities, which make use of acronyms and other specialized language structures (such as medical reports, insurance claims, patient records, etc.).

Acknowledgements. The authors would like to thank the referees for their observations and suggestions, and Devlan Nicholson, Gordon Tisher and the rest of the development team at Axonwave Software for their assistance in accessing and processing the data found within the ASRS database. This research is partially supported by a Discovery Grant from the Natural Science and Engineering Council of Canada.

References

1. The NASA Aviation Reporting System Incident Database, Windows v1.0.6 Data Current Q2 1999, ARG/US Aeroknowledge, Doylestown, PA. (1999).
2. Abney, S.: Part-of-Speech Tagging and Partial Parsing, In Young S. and Bloothooft G. (eds), Corpus-Based Methods in Language and Speech Processing, Chap. 4, pp. 118-136, Kluwer (1997)
3. Church, K.: A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In Procs. 2nd Conference on Applied Natural Language Processing, pp. 136-143, ACL, (1988).
4. Dimitrov, M.: A light-weight approach to coreference resolution for named entities in text. MSc thesis, Department of Information Technologies, University of Sofia: Sofia, Bulgaria. (2002)
5. Lappin, S., Leass, H. J.: An algorithm for pronominal anaphora resolution. Computational Linguistics **20**(4): 535-561 (1994)
6. Mitkov, R.: Robust pronoun resolution with limited knowledge. In Proceedings of COLING'98/ACL'98, Montreal, Quebec, Canada. (1998) 869-875
7. Srinivas, B., Baldwin, B.: Exploiting super tag representation for fast coreference resolution. In Proceedings of the International Conference on NLP+AI/TAL+AI 96, Moncton, NB, Canada. (1996)
8. Terada, A., Tokunaga, T.: Corpus Based Method of Transforming Nominalized Phrases into Clauses for Text Mining Application. IEICE Trans. Inf. & Syst, (**E86-D**)9, (2003).
9. Williams, S., Harvey, M., Preston, K.: Rule-based reference resolution for unrestricted text using part-of-speech tagging and noun phrase parsing. (1996) In Proceedings of the International Colloquium on Discourse Anaphora and Anaphora Resolution (DAARC), Lancaster, UK. (1996) 441-456

Modelling Singularity in Vision to Learn Rotation Invariance toward Recognition

Zou Qi and Siwei Luo

Department of Computer Science, BeiJing Jiaotong University,
Postcode 100044 Beijing, China,
zq089@sina.com
swluo@center.njtu.edu.cn

Abstract. Singularities in visual cortex have been proved to be invariant to orientation. Modelling the mechanism, we explore a recurrent network to realize rotation invariant object recognition. Images sequences as inputs are used to form distinctive features, similar to responses of complex cells. Then recurrent connections decrease distinction of these complex cells leading to emergence of singularities. Invariant features from the singularities acting with memory trace perform object recognition. Memory trace extracts correlations of different views of the same objects from continual sequences, and therefore is fit for performing recognition tasks in visual information processing systems. We testify efficacy of the model by benchmark recognition problem. Our model's plausibility in neurobiology view and practicality in recognition are also discussed.

1 Introduction

A challenging task of what Marr[1] termed *Computational Theory* of object recognition is to generalize recognition across variation of viewpoints. Traditional object recognition needs preprocessing including normalization before matching. For example, we need zoom a face image to specific size, rotate it to frontal orientation beforehand in order to remove difference of perspectives. But neurobiological experiments show vision system may achieve recognition without this preprocess[2]. Some neurons are insensitive to some kinds of changes in visual inputs. This inspires us to build a computational model to achieve object recognition by extracting invariant feature.

Our model differs from previous invariance extraction methods in that it obtains both spatial invariable by topological structure of neural network and temporal correlation by memory mechanism. Using recurrent topological structure to learn invariance originates from recurrent synapses and consequent singularity cells in visual cortex. Temporal correlation, namely intrinsic smoothness of temporal continual sensors, contains invariance in context information. Cooperation of these two mechanism realizes invariant object recognition just as biologic vision does.

In this paper, we'll introduce groundwork in neurobiology briefly, then give our model. Simulation experiments are carried out to demonstrate efficacy of the model and results are analysed. Direction of future work is discussed in conclusion.

2 Evidence in Biology Vision System

Neurobiological experiments prove that neurons in visual cortex V1 area receive afferents from feedforward inputs in a small amount while from recurrent connections of nearby neurons in a large amount[3].The recurrent connections amplify feed- forward inputs. But whether the amplification intensifies or weakens distinctions of neurons has not been well known.

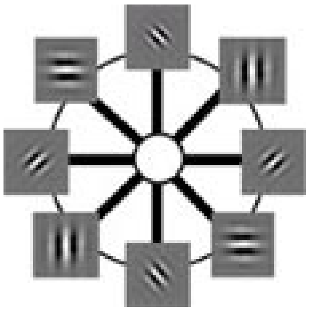


Fig. 1. Ocular dominance column pattern. Cells around the boundary show orientation selectivity. Singularity at the center show orientation invariance as recurrent connections between it and cells around weaken distinctions of orientation selectivities

Psychophysical experiments with human objects show vision system achieves invariant object recognition from different perspectives[4]. Neurobiological experiments with monkeys indicates similar results[2]. As to how visual system obtain these invariances, a principle well accepted by both groups explains that temporal correlation contains invariances of different views of the same object[5,6,7].

Orientation columns in primary visual cortex arrange topographically. Iso-orientation lines intersect the ocular dominance column boundaries at right angles. Neurons near the boundaries exhibit strong orientation selectivities while neurons at the center are insensitive to all orientations known as singularities[3] (see figure 1).

Our model is based on all the mechanism above and try to process visual information in a biological plausible approach.

3 Recurrent Network Model

Figure2 depicts network architecture. It codes stimuli into activity states of cells, which can be expressed by a changed form of classic firing rate model:

$$\frac{dr_i}{dt} = p_i^f(t-1) + p_i^r(t-1) - \langle r_i \rangle_{t-1} . \qquad (1)$$

Changing rate of the *i*th cell's activity state r_i is determined by feedforward input $p_i^f(t-1)$, recurrent input $p_i^r(t-1)$ and $\langle r_i \rangle_{t-1}$ denoting the *i*th cell's memory trace at

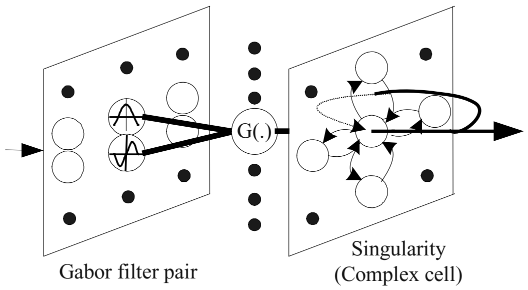


Fig. 2. Recurrent network model (take only one neuron for example). Input is image sequence. Gabor filter pairs extract features as feedforward inputs to singularity. G(.) is operator converting filtered results to complex cell's response(see formula 3).Recurrent connections between singularities are relevant to frequencies. Outputs determining recognition feed back to memory trace and input to singularity at the next time

time instant t-1. It's got by

$$\langle r_i \rangle_t = (1-\mu) r_i(t) + \mu \langle r_i \rangle_{t-1} . \tag{2}$$

where $r_i(t)$ is the *i*th cell's activity state at time instant t ; μ is weight determining influence of past trace and current state. Introduction of memory trace means that contextual information modulates neuronal perception to salient feature. For instance, the same object continually appearing in video is more striking than in static image showed in limited seconds. Formula (1) indicates only if intensity of stimuli at time instant t exceed memory trace at t-1 , the cell can be activated. And only if cells are in continual active states, memory trace can be strengthened so that past memory and current state can be correlated. Features continually activate cells are therefore intrinsic and invariant to objects.

$p_i^f(t)$ is result of stimuli filtered by a pair of Gabor filters $h_i^e(x,y,\theta,f)$ and $h_i^o(x,y,\theta,f)$,consistent with energy function of complex cells in primary visual cortex.

$$p_i^f(t) = G(p_i^e, p_i^o) = \sqrt{(p_i^e(t))^2 + (p_i^o(t))^2} . \tag{3}$$

$$p_i^e(t) = \int_1^M dx \int_1^N dy \{ h_i^e(x,y,\theta,f) \otimes I(x,y,t) \} . \tag{4}$$

$$p_i^o(t) = \int_1^M dx \int_1^N dy \{ h_i^o(x,y,\theta,f) \otimes I(x,y,t) \} . \tag{5}$$

with $h_i^e = e^{-\frac{1}{2}\left(\frac{x'^2}{\sigma_x^2}+\frac{y'^2}{\sigma_y^2}\right)} \cos(2\pi f x')$. and $h_i^o = e^{-\frac{1}{2}\left(\frac{x'^2}{\sigma_x^2}+\frac{y'^2}{\sigma_y^2}\right)} \sin(2\pi f x')$.

where $x' = x\cos\theta + y\sin\theta$. $y' = -x\sin\theta + y\cos\theta$.

$I(x, y, t)$ is the gray value at coordinates (x,y) in $M \times N$ image at time instant t . f and θ is the filter's center frequency and orientation. $\otimes$ is convolution. ${\sigma_x}^2$ and ${\sigma_y}^2$ are bandwidth in two directions.

p_i^r is relevant to cell's selectivity to frequency, but independent of orientation selectivity. Similar frequency selectivities lead to positive p_i^r meaning excitation and different frequency selectivities lead to negative one meaning inhibition.

$$p_i^r(t) = \frac{g_i}{L-1}\sum_{j \neq i}\{2e^{-\frac{(f_i - f_j)^2}{{\sigma_1}^2}} - e^{-\frac{(f_i - f_j)^2}{{\sigma_2}^2}}\}r_j(t) \ . \tag{6}$$

L is the total number of neurons in the network. ${\sigma_1}^2$ and ${\sigma_2}^2$ are frequency tuning bandwidths. $g_i \geq 0$ is the amplification coefficient of recurrent connection. If $g_i = 0$, neurons only receive feedforward inputs filtered by Gabor. As Gabor filters pair resembles receptive field of complex cell[6], neurons show distinctions of complex cells — selective to specific frequency and orientation. When g_i increases, strength of recurrent connections increases. As it is independent of orientation selectivity, recurrent connection between neurons weakens complex cells' distinctions in orientation selectivities so as to produce invariance like singularities. However, too big g_i makes network unstable.

4 Experiment Result

We ran the model on a PC with Pentium 4 CPU and 256M memory. Program tool is Matlab6.5.As the network is built by computation directly rather than by learning iteratively, our model runs in a considerably fast speed. Processing a sequence with seven 64*64images takes about 75 seconds.

We test the model's ability to extract orientation invariance on 64*64 plane images showed in figure3. We get five classes with each comprising a sequence of length 7. Each sequence is 7 different views of the same object, i.e. planes at 7 different angles uniformly distributed between $-\pi/2$ and $\pi/2$. Take output of a plane at preferred angle as reference value, then responses to rotated stimuli are expressed as percent of reference values. Derivative in formula (1) is approximated by $dr_i/dt = r_i(t) - r_i(t-1)$. We appoint ${\sigma_1}^2$ =0.25 , ${\sigma_2}^2$ =1 presumed from bandwidth constant. Frequencies of Gabor filters are 2^{-2} , 2^{-1} and 2^0 . Orientations are $k\pi/8$, $k \in \{0,...,7\}$. So total 48 Gabor filters process inputs and total 24 neurons in the network produce feature vectors whose component represents feature at specific frequency and orientation. For instance, the plane tu-26 in view like the third image in figure3 shows most salient features at frequency of 2^{-2} and 2^{-1} and orientation of 0^o .The neuron corresponding to frequency of 2^{-2} and orientation of 0^o as well as the one corresponding to frequency of 2^{-1} and orientation of 0^{o} response most strongly. And these two neurons response relative strongly to plane tu-26 in rotated views because of recurrent connections and memory trace.

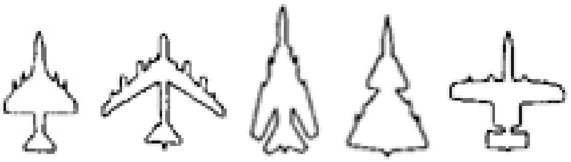


Fig. 3. Plane image from H. Kim's database in North Carolina State University. From left to right: a-4,b-52,tu-26,viggen and a-10a

Figure4 shows two typical singularity responses to rotated stimuli with rotation range between $-\pi/2$ and $\pi/2$. As we can see, the model extracts stable invariance. Besides, the neuron responses strongly to one class while hardly to others. This reflects the model can discriminate different classes.

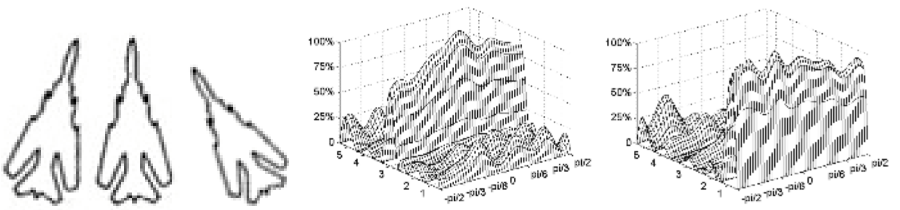


Fig. 4. Rotation invariance. *Left* : reference image and rotated images (only take two frames for example). *Right* : two singularity response curves. *X* axis is rotation angles, *y* axis is different object classes, *z* axis is responses to specific object at specific angle

In order to measure recognition ability quantitively, we define difference between object classes *Dif(p,q)* = $\arccos\left(< v_p, v_q > /(\| v_p \| \| v_q \|)\right)$ where v_p is feature vector of input object *p*, $<,>$ is inner product, is Euclid norm. Each object rotated to various angles produces images sequences. When rotated object has the least difference with reference value of the same class of object, they are classified correctly, otherwise are wrong classified. We investigate performance of the model under different conditions.

When μ is 0.6 and g_i is 0.82, performance is wonderful and correct recognition ratio is 85.6%. When μ is 0, namely memory doesn't exit, recognition ratio is 21.6%. The model loses recognition ability as information of time correlation disappears. So different objects at the same angle may be judged more similar than the same object at different angles. When g_i is 0,namely recurrent connections doesn't exit, the ratio is 30.1%. The model can't extract rotation invariance. So the same object at different angles may be judged as different objects. We can see memory trace and recurrent connections determine invariant recognition ability of the model.

We compare performance of our model to others which obtain rotation invariance by exploiting temporal correlation. Wiskott applied SFA(slow feature analysis)[7] to one dimension Gaussian signals to learn invariance. Under the same criterion as defined in our experiment, his accuracy is 88%. But his network has more complex hierarchy and more complex learning procedure while experiment on more simple objects due to constraint of receptive fields size. Chance[8] also use recurrent network to learn invariance. But his model only learned phase invariance of analog signals like drifting gratings. And his purpose was to learn properties of complex cells rather than applying in practical recognition task.

5 Conclusion

The paper produces singularity using recurrent network so as to extract rotation invariance. Memory trace acts with invariant feature to gain object recognition. It provides a way to model vision neurons response property. While receptive fields like Gabor filter extract selective features as complex cells do, recurrent connections help extract invariant features as singularities do. Besides, the model is implemented in a neurobiologic plausible way in several other aspects such as exploiting temporal correlation and limited feedback connections. We test the model with binary image of simple objects. Extending it to gray images gets similar results. We do this for the purpose of saving computation cost. The point is that biology vision inspiring model can gain acceptable results with high speed in even simple way. Extending it to more complex objects and larger number of classes are our future work. Besides, selection of parameter values needs strict methodical analysis rather than by experience.

We believe with deeper research in biology vision, using large scale parallel computation model and combining other vision mechanism such as attention, developing intelligent recognition system is possible.

References

1. Marr, D.: Vision. W.H. Freeman and company, San Francisco, USA (1982)
2. Hasselmo, M.E., Rolls, E., Baylis, G.: The Role of Expression and Identity in the Face-selective Responses of Neurons in the Temporal Visual Cortex of the Monkey. Behav. Brain Res. Vol.32. (1989) 203-218
3. Peters, A., Payne, B. R.: A Numerical Analysis of the Geniculocortical Input to Striate Cortex in the Monkey. Cereb. Cortex. Vol.4. (1994) 215–229
4. Furmanski, C. S., S. A. Engel: Perceptual Learning in Object Recognition: Object Specificity and Size Invariance. *Vision-Research.-40-*(5) (2000) 473-484
5. Fordiak, P.: Learning Invariance from Transformation Sequences. Neural Computation. 3(2), (1991) 194-200
6. Morrone, M. C., Burr, D.C.: Feature Detection in Human Vision: a phase-dependent energy model. proc. Royal soc. London ser. B, 235(1280). (1988) 221-245
7. Wiskott, L., Sejnowski, T.: Slow Feature Analysis: Unsupervised learning of Invariances. Neural Computation.Vol.14. (2003) 715-770
8. Frances S. Chance, Sacha B. Nelson, L.F. Abbott: Complex Cells as Cortically Amplified Simple Cells. Nature neuroscience. 2(3) (1999) 277-282

Two Set-Theoretic Approaches to the Semantics of Adjective-Noun Combinations

Nabil Abdullah and Richard Frost

School of Computer Science, University of Windsor, Windsor, Ontario ON N9B3P4.
{nabil_abdullah@yahoo.com, richard@uwindsor.ca}

Despite their simple syntactic form, adjective-noun combinations seem to have no straightforward semantic method that parallels the simplicity of the syntax. Conventionally, adjectives are to blame for this behaviour. Such a belief has culminated in the generally accepted view that adjectives should be semantically analyzed as belonging to a hierarchy of the following structure [1, 2]:

- Intersective e.g. *red* as in *That is a red rose*
- Subsective:
  - Subsective-only e.g. *veteran* as in *John is a veteran soldier*
  - Double or "doublet" e.g. *beautiful* as in *Maria is a beautiful dancer*
- Non-subsective:
  - Non-privative e.g. *potential* as in *John is a potential candidate*
  - Privative e.g. *fake* as in *That is a fake gun*

Each node in the hierarchy requires a different semantic rule for calculating the meaning of the compound involving the relevant adjective. Furthermore, compounds with constituent adjectives such as 'beautiful' require two different rules for the same compound. This apparently "undisciplined" semantic behaviour of the single syntactic group "adjective" has been used by some authors to further the thesis of non-compositionality of natural-language expressions. Contrary to such views, we believe that adjectives are more systematic in their behaviour than originally thought. In support of this claim, we propose two set-theoretic approaches to the semantics of adjective-noun combinations. The first hypothesizes that an adjective-noun compound is a subset of its constituent noun. The second hypothesizes that the adjective-noun combinations can semantically be thought of as set intersection involving the adjective(s) and the head noun of the compound. In both approaches, we exclude adjectives with modal import, i.e. "non-privative" such as 'potential' and 'possible', arguing that such adjectives are better be treated within modal logic.

An essential step towards a compositionally uniform treatment of adjective-noun combinations is our view that the class of adjectives known as "privative" can be accommodated within an existing class in the adjective hierarchy, known as "subsective". This is achieved by "augmenting" the domain of common nouns. For example, 'gun' has fake and real guns in its domain and 'president' has former and current presidents in its domain. Consequently, the compound *fake gun* in the sentence *That is a fake gun* becomes subsective on par with expressions such as *ultimate gun*.

Since the adjective hierarchy is now reduced to intersective and subsective classes, a generalized solution to the semantics of adjective-noun combinations is attainable. In our first approach we view all adjective-noun combinations as adhering to the schema, where the symbol "|| ||" represents the denotation of the expression.

$$\text{for all adjective } A \text{ and noun } N, \quad \| A\ N \| \subseteq \| N \|$$

In special cases the expression $\| A\ N \| \subseteq \| A \|$ is also true, e.g. colour adjectives.

The interpretation function of an adjective-noun combination can be computed as follows. Let *an* be a node in a parsing tree dominating the nodes *adj* and *noun*. Then the meaning of *an* is defined as follows:

$$\|an\| = F\,(\,\|adj\|, \|noun\|)$$

where $\|noun\|$ is a set and $\|adj\|$ is the function *g* if *adj* is subsective, otherwise, $\|adj\|$ is a set. *F* and its composite function *g* are defined as follows:

$$F \begin{cases} g\colon \|noun\| \rightarrow \|noun\| & \text{if } adj \text{ is subsective} \\ \|adj\| \cap \|noun\| & \text{if } adj \text{ is intersective} \end{cases}$$

It is evident that *F* is a function from a domain of a noun of a compound to a subset of it. It should be noted that in our implementation *g* is realized by enumerating the relevant subset and composing an appropriate name for it.

In the second approach, the ontological difference between nouns and adjectives is encoded in their respective representation by means of type-endowed sets. Using such sets, adjectives and common nouns have, respectively, the following representation schemata:

Adj_set = {member: type1}: type2

Noun_set = {member: type}: type

We show that such a representation is capable of producing consistent inferences involving all kind of adjective-noun combinations. In addition, we demonstrate how this approach is capable of computing the meaning of compounds with multiple adjectives and a head noun such as *fake skilful dancer* in *Jane is fake skilful dancer*. Finally, we show that this representation eliminates the co-extension problem without resorting to notions such as "possible worlds". The interpretation function *F* in this case is much simpler. It only involves set-intersection: i.e. $F\,(\,\|adj\|, \|noun\|) = \|adj\| \cap \|noun\|$.

The two approaches are alternatives. As such they have advantages and disadvantages. However, they have two common advantages. First, the single syntactic group "adjective" has a single semantic rule. This is more evident in the second approach. We believe this is an important result which advocates the thesis that natural-language expressions are compositional. Second, both are machine tractable.

Acknowledgements. The authors acknowledge the support of NSERC.

References

[1] Kamp, H. and Partee P.H.(1995) Prototype theory and compositionality. *Cognition* 57 pp 129-191.

[2] Siegel, M. (1976) *Capturing the adjective*. PhD dissertation. The University of Massachusetts at Amherst.

Comparison of Permutation-Based and Binary Representation in a Genetic Algorithm for RNA Secondary Structure Prediction

Alain Deschênes, Kay C. Wiese, and Edward Glen

Simon Fraser University, Information Technology,
2400 Central City, 10153 King George Highway,
Surrey, B.C., V3T 2W1, CANADA
{aadesche, wiese, and eglen}@sfu.ca

RNA is an important molecule as it serves a key role in the translation from the genetic information encoded in DNA in protein synthesis. Computational techniques for RNA folding suffer from combinatorial explosion. In this paper, a genetic algorithm (GA) will be used to attempt to solve the secondary structure prediction of RNA molecules.

The current paper will: 1) Demonstrate the benefits of using permutation-based encoding over binary encoding by a direct comparison of the two algorithms, 2) Compare the relative merits of different permutation-based crossover operators in the RNA folding domain, and 3) Demonstrate the effect of Keep-Best Reproduction[1] (KBR) as compared to Standard Selection (STDS).

In RNA secondary structure prediction, the GA attempts to find low energy, stable structures within a given thermodynamic model. Since a base pair does not form in isolation in our model, it is useful to find the set of all possible helices H. A valid secondary structure is the subset S of H containing the helices that make up the actual structure. This problem is highly combinatorial as there are $2^{|H|}$ subsets of H.

We have tested RNA sequences of different lengths over a wide variety of operators and parameter settings. We varied the replacement strategy using both STDS and KBR. STDS selection involves selecting parents from a standard roulette wheel selection where the offspring are inserted in the next generation, while KBR also selects parents via roulette wheel but after crossover, the best parent and the best offspring are inserted in the next generation. We also compare the results of the permutation-based GA with a binary GA. For the binary based encoding, we used 1-point, 2-point, and uniform crossover operators. In the permutation-based GA, the crossover operators used were: CX[2], PMX[3], OX[4], SYMERC[5], and ASERC[6].

When comparing different encodings and crossover operators using STDS (Figure 1), we see that none of the binary runs make progress toward lower energy structures. However, permutation-based operators used with STDS show an improvement where CX is found to be the best operator with an improvement of 13% over the initial random population's best structure. The other less performant operators improved on the initial population's best structure by 5.9%, 5.0%, 4.2%, 3.9%, 0.9%, 0.4%, and 0.02% for PMX, OX, ASERC, SYMERC, 1-point, 2-point, and uniform, respectively.

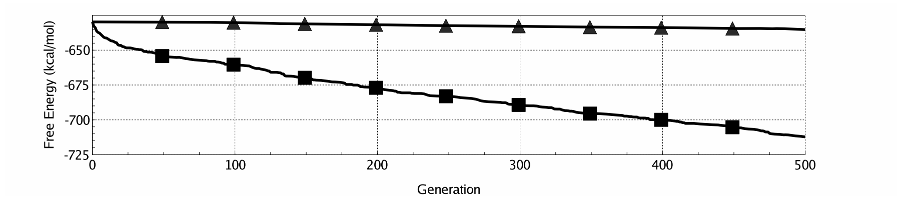


Fig. 1. This graph is a comparison of our best permutation-based operator, CX (squares), and our best binary operator, 1-point (triangles), using STDS. The series are from 30 random seeds, averaged, using a 785 nucleotides sequence. The CX operator converges at a higher velocity than the 1-point operator.

When using KBR, both algorithms made substantial progress. Initially, CX, our best permutation-based operator, converges at the highest velocity only to be outperformed by a slight margin at the last few generations by the uniform binary crossover. Both these operators achieve an improvement of 17.3% over the initial random population's best structure. These crossover operator are followed by PMX, 2-point, 1-point, ASERC, SYMERC, and OX with improvements over the initial population's best structure of 17.0%, 16.8%, 16.6%, 14.4%, 13.9%, and 12.63%, respectively.

In conclusion, our results clearly indicate a benefit of using KBR as opposed to STDS. Any experiment conducted with KBR enabled outperformed all STDS experiments. Also, under most condition, permutation-based encodings outperform binary based encodings.

References

1. K. C. Wiese and S. D. Goodwin, "Keep-Best Reproduction: A Local Family Competition Selection Strategy and the Environment it Flourishes in," *Constraints*, vol. 6, pp. 399-422, 2001.
2. I. M. Oliver, D.J. Smith, and J. R. C. Holland, "A Study of Permutation Crossover Operators On The Traveling Salesman Problem," in *Proceedings of the Second International Conference on Genetic Algorithms*, 1987.
3. D.E. Goldberg, R. Lingle, "Alleles, Loci, and the Traveling Salesman Problem," in *Proceedings of the International Conference on Genetic Algorithms and their Applications*, 1985.
4. L. Davis, "Job shop scheduling with genetic algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms ICGA-85*, 136-140, 1985.
5. D. Whitley, T. Starkweather, and D. Shaner, "The travelling salesman and sequence scheduling: quality solutions using genetic edge recombination", in *Handbook of Genetic Algorithms*, L. Davis, Ed., London: International Thomson Computer Press, 1996, pp. 350–372.
6. K. Wiese, S. D. Goodwin, and S. Nagarajan, "ASERC - A Genetic Sequencing Operator for Asymmetric Permutation Problems,", *Lecture Notes in Artificial Intelligence - Advances in Artificial Intelligence*, vol. 1822, H. Hamilton and Q. Yang, Eds., pp. 201-213, 2000.

Time-Sensitive Sampling for Spam Filtering

Ching-Lung Fu and Daniel Silver

Jodrey School of Computer Science, Acadia University, NS, B4P 2R6
{032790f,danny.silver}@acadiau.ca

Abstract. Email filters based on learned models should be developed from appropriate training and test sets. A k-fold cross-validation is commonly presented in the literature as a method of mixing old and new messages to produce these data sets. We show that this results in overly optimistic estimates of the email filter's accuracy in classifying future messages because the training set has a higher probability of containing messages that are similar to those in the test set. We propose a method that preserves the chronology of the email messages in the data sets.

1 Introduction

In our research into email spam filtering, we have found that the accuracy of existing filtering systems based on learned models is overly optimistic [1]. It is difficult to create models with the same level of high accuracy as published by the authors when tested on independent data sets. Much of the variance may be attributed to differences among the datasets. We speculated that another important factor is the testing method used by the authors. Our work on spam filtering has shown that time is an important factor for valid testing; *i.e.* the order of incoming email messages must be preserved to properly test a model. Many results reported on spam filtering are based on a k-fold cross-validation methodology that does not preserve the temporal nature of email messages. Unfortunately, we could not find previous literature discussing this temporal issue. It is common practice with cross-validation to mix the available data prior to sampling training and test sets to ensure a fair distribution of legitimate and spam messages [2]. This mixes older messages with more recent and generates training sets that unfairly contain the later messages. A fair training set should never contain any messages received after those used to develop the model. Commonly used datasets available on the web, such as the Ling-Spam corpus [3], do not even contain a time stamp in the data. If this temporal aspect is not considered, the performance of the model on future predictions will likely be less than that estimated. We present a comparison of a spam filtering system tested using cross-validation and a temporal preserving approach.

2 Theory

We propose that mixing the available email messages prior to choosing the training and test sets will produce overly optimistic results; that is, the model will not perform as well on future messages as it does on the test set. When the data sets are mixed, possible future examples are injected into the training set; therefore, providing the model with an unfair look at future examples. We propose that a more realistic evaluation is to select the training and test sets while the data is in chronological order. The test set will then simulate future messages which can be classified based on a model produced from prior messages. In this way the test set accuracy should better estimate the performance of the spam filter.

3 Experimentation

Method and Approach: The purpose of the experiment is to compare a k-fold cross-validation approach that does not preserve the chronology of email messages with a data sampling approach that does. A back-propagation neural network was used to develop our spam filter models. The email data set was collected from a single individual from January to May, 2003, and consists of 1000 messages sorted chronologically. The spam and legitimate messages have equal proportion. A block of 500 messages were chosen for each repetition of the experiment starting at the earliest email message. On each repetition the block was moved 100 messages forward in the chronology. From each block of messages, 300 were selected for a training set, 100 for a tuning set and 100 for a test set. Prior to model development, features were extracted from each message using traditional information retrieval methods; *i.e.* removing stop words, performing word stemming, and collecting word statistics. Two methods of data sampling were used to select the training, tuning and test sets. For Method 1, the message data for the three sets was randomly chosen (as per standard cross-validation). For Method 2, the most recent messages were selected for the test set and the remaining messages randomly distributed to the training and tuning sets. The tuning set was used to prevent over-fitting of the neural network to the training set. Six repetitions of each method were completed.

Results

Table 1. Mean accuracy and difference of mean results

	Method 1	**Method 2**	***p*-value**
Mean Accuracy	0.930	0.905	0.00675

Table 2. Detail statistics and difference of means results

	Method 1				**Method 2**			
	TP	TN	FP	FN	TP	TN	FP	FN
Mean	0.88	0.98	0.11	0.02	0.88	0.93	0.12	0.07
				***p*-value**	0.71876	0.02653	0.71876	0.02653

4 Discussion

The results shown in Table 1 indicate that Method 2, which ensures the test set contains the most recent email messages, has a mean accuracy that is significantly less (by 2.5%) than Method 1, which uses a test set with a mixed chronology. Method 1 over-estimates its performance on future messages because the test set has a higher probability of containing messages that are similar to those in the training set. Method 2 generates a less accurate but more realistic model because the testing procedure simulates the classification of chronologically correct in-coming messages. Table 2 shows at a finer level of assessment that Method 1 is significantly better at classifying spam messages (True Negatives = TN). Our conjecture is that Method 1 unrealistically allows the modeling system to identify features of future spam emails where Method 2 does not. The difference in the TN statistic between the two methods may not appear to be important as it only represents 5 misclassified spam emails out of 100. However, if the same model were used over a longer period of time the effect would increase.

References

[1] I. Androutsopoulos *et al.* 2000. Learning to Filter Spam E-Mail: A Comparison of a Naive Bayesian and a Memory-Based Approach. *Proceedings of the workshop on Machine Learning and Textual Information Access", 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-2000)*, H. Zaragoza, P. Gallinari and M. Rajman (Eds.), Lyon, France, Sept, 2000, p. 1-13.

[2] T. Mitchell. *Machine Learning*. McGraw Hill, New York, NY, 1997.

[3] I. Androutsopoulos homepage, http://www.aueb.gr/users/ion/publications.html

Comparison of Parallel and Serial Genetic Algorithms for RNA Secondary Structure Prediction

Andrew Hendriks, Kay C. Wiese, and Edward Glen

[1] Simon Fraser University, Information Technology,
2400 Central City, 10153 King George Highway
Surrey, B.C., V3T 2W1, CANADA
Phone: (604) 268-7412, Fax: (604) 268-7488
{ahendrik, wiese, eglen}@sfu.ca

The basic function of a biomolecule is determined by its 3 dimensional shape, otherwise known as the tertiary structure. However, existing empirical methods to determine this shape are too costly and lengthy to be practical. RNA is of interest as a biomolecule because it is central in several stages of protein synthesis. Also, its secondary structure dominates its tertiary structure. In our model, RNA secondary structure develops as a consequence of bonds which form between specific pairs of nucleotides known as the canonical base pairs. Searching a sequence of nucleotides for all possible base pairs is rapid and straightforward; the challenge comes from attempting to predict which specific canonical base pairs will form bonds in the real structure. Various algorithms have been used for RNA structure prediction such as dynamic programming, and comparative methods [1] and stochastic methods such as genetic algorithms (GA).

The fitness metric employed to guide a given GA through the search space of possible base pairs is energy minimization. As an RNA molecule will fold into a structure with near minimal free energy (FE) [2], the GA attempts to find the structure resulting in the lowest energy possible. Wiese and Glen (2002) noticed promising performance when using a permutation-based GA, especially when employing Keep Best Reproduction (KBR) techniques, Cycle crossover (CX) and 1-elitism [3]. This suggested the possibility of further improvement through the incorporation of a coarse-grained, distributed GA model.

Coarse-grained distributed GAs offer a speed up through parallelization, the prevention of premature convergence by maintaining diversity, an increase of the selection pressure within the entire population, and also a reduction of the time to convergence [4].

A series of tests were performed using a simulated distribution on a series of sequences, with a wide variety of parameters. The rate of convergence is accelerated in the distributed GA, which can likely be attributed to increased selection pressure from elitism per deme and periodic migration; this is more pronounced with Standard Roulette Wheel Selection (STDS), as KBR's selection pressure is already heightened.

The best distributed STDS run showed an improvement of 14%, versus the serial run at 12%. The distributed and serial KBR runs had nearly identical percentage improvement.

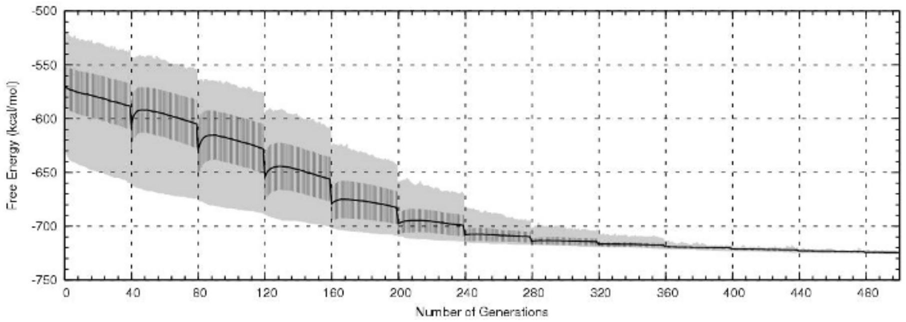


Fig. 1. A plot of the best distributed GA run employing STDS, CX, deme size = 100, deme count = 7, Pc = 70, Pm = 5, 1-Elitism, migration interval of 40 and rate of 10

In Figure 1, sharp cyclic deviations in the mean (black line) and bounding dark-grey standard deviation of the fitness are visible. The fitness cycles are clearly the result of the migration occurring every 40 generations; the minimum fitness improves if diversity is still present. The children produced by crossover with the new migrants in the next generation account for the sudden drop in the maximum (worst) FE.

In summary, the rate of convergence is substantially accelerated in the distributed GA, which can likely be attributed to increased selection pressure from elitism per deme and periodic migration; this is more pronounced with STDS, as KBR's selection pressure is already heightened. Overall, we can achieve swifter convergence and comparable quality of results in terms of FE to the serial GA, especially when employing STDS replacement.

References

1. Woese, C. and Pace. N.: Probing RNA structure, function, and history by comparative analysis. In the RNA World, Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY.(1993)
2. Mathews, D.H., Andre, T.C., Kim, J., Turner, D.H., and Zuker, M.: An Updated Recursive Algorithm for RNA Secondary Structure Prediction with Improved Free Energy Parameters. In Molecular Modeling of Nucleic Acids, Leontis, N. B.; SantaLucia, J. Jr., editors. American Chemical Society, pp. 246-257 (1998)
3. Wiese, K. C. and Glen, E.: A Permutation Based Genetic Algorithm for RNA Secondary Structure Prediction, Frontiers in Artificial Intelligence and Applications, Vol. 87, Soft Computing Systems, IOS Press, pp. 173 - 182.
4. Cantú-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Netherlands: Kluwer Academic Publishers (2000)

Performance Evaluation of Agent Toolkits

Yang Jun and Elhadi Shakshuki

Computer Science Department
Acadia University
Nova Scotia, Canada B4P 2R6
{052982y;elhadi.shakshuki}@acadiau.ca

Abstract. There are many Agent Building Toolkits to develop multi-agent systems (MAS) in many businesses and in research areas. This makes it difficult for the developers to select the appropriate one. Thus, it is necessary to compare these toolkits to help developers to choose the most appropriate tool for their desired applications. This paper presents an evaluation of Java Agent Development (Jade) framework, Zeus Agent Building Toolkit (Zeus) and JACK Intelligent System (Jack). The evaluation is focused on performance evaluation on message transport system of Agent Building Toolkits.

1 Introduction

Currently there are many agent building toolkits to develop MASs in business or research areas. Using these toolkits facilitates the process of developing such systems quickly and efficiently. But, there exist a large number of toolkits [2], which makes it difficult to select an appropriate one. There have been few attempts by researchers to compare these toolkits. The EvalAgents project aimed at evaluating the agent building toolkits on Java support, mobility, security and development support areas [3]. In another direction, Camacho et al. [4] performed experiments to test and compare the performance on Zeus, Jade and Skeleton-agent frameworks, using the meta-search engine workload. Although they performed useful experiments, there was little work done on the performance measure of the message transport system (MTS) of toolkits. One of the main key components of the toolkits is the MTS. The second most important and challenging issue in MASs is the transport layer for agent communication [1]. Towards this end, we developed a benchmark to compare Jade, Zeus and Jack agent building toolkits performance on the MTS. We implemented a prototype using each of these toolkits to measure their performance. The evaluations of these toolkits were performed on a notebook pc with Intel Pentium 4 1.6 GHz processor, 256 MB RAM with a JDK 1.4.0 running Windows XP.

2 Comparisons and Results

This section provides a comparison between the three agent building toolkits, including Jack, Zeus and Jade. First, all of these toolkits provide Java support. Jade and Zeus are available free of charge. Jack is provided as a trial version for 60 days. Zeus and Jade are FIPA-compliant agent platforms. Jack does not make use of any pre-existing standard agent communication language. Jade uses an agent model and

Java implementation that offer good runtime efficiency and software reuse. Conversely, Zeus offers very powerful means to graphically design MASs, with the desired interactions. Jack provides an excellent GUI for defining agents within projects. Jack includes all components of the Java development environment as well as offering specific extensions to implement agent behaviour. In our experiments, two cases are considered. In the first case, the communication between agents is established when both agents are living in the same container. In the second case, the communication between agents is established when agents are living in different containers. The proposed performance matrix consists of two fields, namely: the number of active agents and the average round trip time. Each JVM is the basic container of agents that provides a complete run time environment for agent execution and allows several agents to concurrently execute their tasks on the same host. Fig. 1 shows that Jade provides better performance on the MTS than Jack and Zeus when the agents live in the same container.

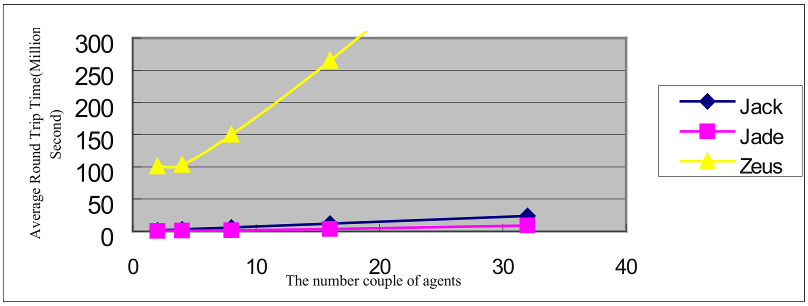


Fig. 1. Performance results for Jade, Jack and Zeus on one container of MTS.

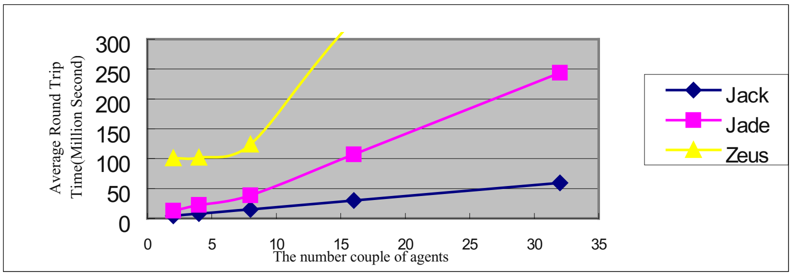


Fig. 2. Performance results for Jade, Jack and Zeus on two containers of MTS.

Fig. 2 shows that Jack provides better performance on the MTS than Jade and Zeus when the agents live in different containers.

3 Conclusion and Future Work

This paper compared Jade, Zeus and Jack multi-agent toolkits and proposed a benchmark to evaluate them. The agent building toolkits provide their own architecture and build-up methodology to deploy MASs and have different performances. Based on our investigations, it is recommended that Jade can be used when the application requires agents to live in the same container, because it provides better performance. Alternatively, it is recommended to use Jack when the application requires agents to live in different containers.

In the future, we will continue to work on other agent building toolkits, using the same test measurements as well as incorporating other important criteria. Furthermore, we will design and implement user friendly interfaces that allow developers to find the most appropriate toolkits based on their required needs.

References

[1] Fonseca, S., Griss, M. and Letsinger, R., ‘‘Evaluation of the Zeus MAS Framework’’, Software Technology Laboratory HP Laboratories Palo Alto, HP Labs Technical Report, HPL-2001-154, 2001.

[2] MultiAgent Systems, Tools for building MASs. http:// www.multiagent.com/Software/

[3] Grabner, M.,Gruber,F., Klug,L.,Stockner,W., Altmann,J. and Essmayr,W., “Technical Reprot: SCCH-TR-00-64-1: EvalAgents: Evaluation Report”, 2000.

[4] Camacho, D., Aler,R., Castro,C. and Molina.M.J, “Performance Evaluation of Zeus, Jade, and Skeletonagent Frameworks”, 2002.

Software Agents in CVW

Ming Zhen Lei, Elhadi Shakshuki, and Ivan Tomek

Jodrey School of Computer Science
Acadia University
Wolfville, Nova Scotia, Canada B4P 2R6
{0607111, elhadi.shakshuki, ivan.tomek}@acadiau.ca

Abstract. This paper reports on ongoing research on developing several types of software agents including observer agents, statistical agents, scheduling agents, tutoring agents, training agents, and identifying agents for the Collaborative Virtual Workspace (CVW). We present the architecture, design and implementation of the observer agent whose tasks are to monitor and report events related to user activities and document-related events in CVW.

1 Introduction

This paper reports on a part of our CVE-related research, the development of a systematic approach to building software agents in Virtual Environments. A text-based Collaborative Virtual Environment (CVE) is a GUI-based virtual reality in which users proxies move among virtual places, communicating, and manipulating their environment [1, 2]. The test bed of our work is CVW, an open-source CVE developed by MITRE [3]. This paper focuses on augmenting CVW with software agents, programs that perform information gathering in the background [4].

This paper discusses an observer agent's architecture for CVW running on a CVW client. Its main function is to assist users in monitoring events related to selected activities of other users and documents, allowing the users to customize their needs such as the names of targeted users. The agent provides alert messages when a user logs in or logs out, as well as monitor and report changes to the document base such as document modification or deletion.

2 System Architecture

CVW uses client/server architecture to implement shared virtual space. Its core components are the server, the document server, and the client. The client and server maintain persistent connections and communicate via TCP/IP using the MOO Command Protocol (MCP) [5].

Our observer agents run on CVW client side, as shown in Fig. 1-a. They consist of five main components: communication, interaction, knowledge base, problem solver, filter and execution unit, as shown in Fig. 1-b. We will now explain the main components briefly.

The ability to *communicate* allows agents to send and receive MCP commands related to CVW components, including users and documents. The *interaction* component provides a user interface to configure the agent's settings. The *knowledge base* stores information that the observer agent needs to achieve its goals and includes user preferences and parameters such as user and document events to be monitored and used for notification. Users can specify their parameters and preferences through a graphical user interface and save them in the knowledge base. The *problem solver* provides the agent with the ability to reason about its knowledge base and generate solutions to satisfy user goals. The *filter* component is responsible for monitoring events occurring in the user's current room. It detects events that correspond to users' interests, concerns and preferences and sends the relevant information to the problem solver. Examples of such MCP commands are #$#cvw-user-connection-notify and #$#cvw-user-move-notify. The *execution component* carries out the solution that generated by the problem solver.

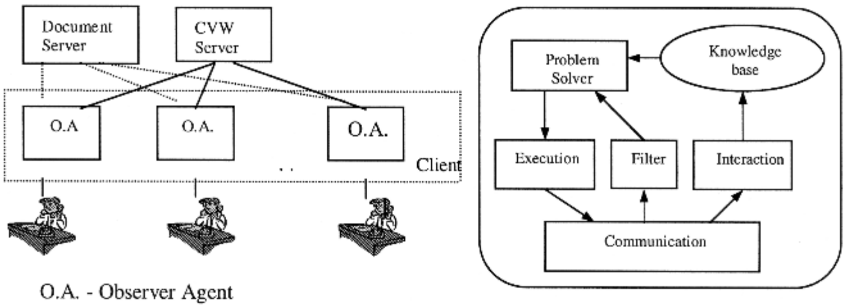
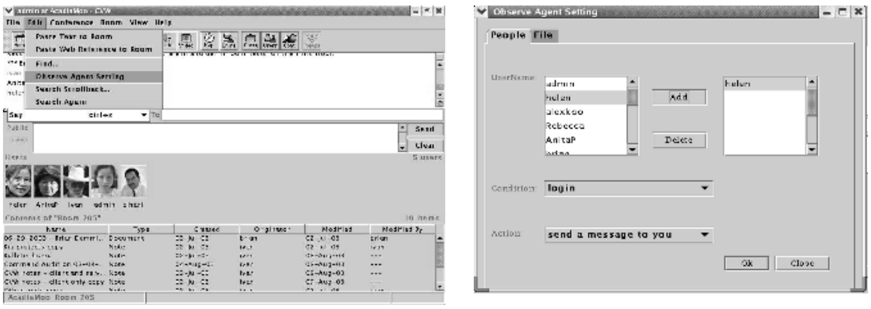


Fig. 1. (a) System Architecture and (b) Observer Agent Architecture.

3 Implementation

Observer agents are realized by extending the open source JAVA implementation of the CVW client software, using the interface shown in Fig. 2-a, and employing MCP commands for communication between the client and the server. The GUI shown in Fig. 2-b allows the users to specify the agents' properties.



UserName: admin, helen, alexkao, Rebecca, AnitaP — helen

Condition: login

Action: send a message to you

Fig. 2. (a) CVW client interface and (b) Observer agent interface.

4 Conclusion and Future Work

This paper presented the architecture and the implementation of an observer agent in CVW. This agent helps users to monitor events related to selected activities of other users and documents, allowing the users to customize their needs. The agent permits users to specify the names of targeted users and provide alert messages when a user logs in or logs out, as well as monitor and report changes to the document base such as document modification or deletion. The work presented in this paper described an experiment with a client-side agent, a first step in our exploration of the possible implementation of agents in virtual e-Learning environments. Although client-side agents have their place in our target environment, server-side agents inhabiting the CVW virtual universe and supporting our goals by direct interaction with user proxies, objects, and the virtual space are even more important. In our future work, we will develop server-side agents specifically aiming at supporting educational tasks.

References

[1] Churchill, E., Bly S., It's all in the words: Supporting work activities with lightweight tools, Proceedings of Group '99, 1999.

[2] Haynes, C., Holmevik, J. R. High wired: On the design, use, and theory of educational MOOs. University of Michigan Press, 1998.

[3] CVW, Collaborative Virtual Workspace, http://cvw.sourceforge.net.

[4] Webopedia: Online Dictionary for Computer and Internet Terms. http://www.webopedia.com/.

[5] http://cvw.sourceforge.net/cvw/info/mcpdocs/protocols/current_protocols.php3, 2001.

Constraint Directed Dynamic Backtracking

Eric J. Mulvaney and Scott D. Goodwin

University of Windsor, Windsor, ON, N9B 3P4, Canada
{mulvane;sgoodwin}@uwindsor.ca

Introduction

A wide range of problems can be represented as Constraint Satisfaction Problems (CSPs). A simple backtracking (BT) search algorithm can be used to solve CSPs, but efficiency is a concern. Pang's constraint directed backtracking (CDBT) algorithm [2] is another method for solving CSPs. This algorithm attempts to improve search efficiency by using the constraints to guide the search. One way to understand the CDBT algorithm is by viewing it as BT in an alternative (dual) representation of CSPs. In the dual representation, the dual variables are the original (primal) constraints, the domains of the dual variables are the tuples satisfying their corresponding primal constraints, and the dual constraints (called compatibility constraints) restrict the values assigned to dual variables in such a way that the implied values of the primal variables shared by the corresponding primal constraints are consistently assigned. While it is known that CDBT can outperform BT, both algorithms suffer from the problem of thrashing. Many improvements to reduce thrashing in BT have been studied. Ginsberg's dynamic backtracking (DBT) algorithm [1] attempts to reduce thrashing by backtracking over variables not involved in a conflict, while preserving their values. Here we investigate whether DBT can be adapted to CDBT to address thrashing. We call our proposed hybrid algorithm Constraint Directed Dynamic Backtracking (CD-DBT).

Definitions

Suppose we are given a CSP, where V is a set of variables, D is a set of domains, and C is a set of constraints. Each $v \in V$ can be *undefined* or *assigned* a value from its domain D_v. Each $c = (V_c, S_c) \in C$ *constrains* $V_c \subseteq V$ and is *satisfied* iff the variables in V_c are assigned values from a tuple $s \in S_c$.

A solution $s \in S_c$ is *prohibited* by $\mathcal{C} \subseteq C - \{c\}$, iff for every $c' \in \mathcal{C}$ there exists at least one variable $v \in V_c \cap V_{c'}$ assigned a value different from the value needed by the solution s. The values $s \in S_c$ may be *assigned*, if s is not prohibited; if assigned, every undefined variable $v \in V_c$ is assigned a value from s. If the values $s \in S_c$ are *unassigned*, every variable $v \in V_c$ is undefined iff it is not required by any satisfied constraint.

Dynamic Backtracking

DBT [1] begins with all variables undefined and an empty set of eliminations for each variable, $E_{v \in V} = \varnothing$. The elimination set consists of tuples $(x, \mathcal{V}) \in E_v$, where $x \in D_v$ cannot be assigned to v, because of the values currently assigned to the variables $\mathcal{V} \subseteq V$.

It then proceeds as follows: (Step 1) If all variables are assigned, *a solution has been found*, otherwise pick an undefined variable v. Update E_v considering the variables assigned and the values not already eliminated by E_v – keep old eliminations. (Step 2) If at least one value $x \in D_v$ is not excluded by E_v, assign x to v and go to Step 1. (Step 3) Let v' be the most recently assigned variable, which is also in E_v. If v' cannot be found, *no solution exists*. Otherwise, let x' be the value assigned to v', then undefine v' and remove every $(x, \mathcal{V})$ from every $E_{v \in V}$ when $v' \in \mathcal{V}$. Add $(x', \mathcal{V}_{E_v})$ to $E_{v'}$, where $\mathcal{V}_{E_v}$ is the set of all variables in E_v, and go to Step 1.

Constraint Directed Dynamic Backtracking

CD-DBT begins with all the variables unassigned and with an empty set of eliminations for each *constraint*, $E_{c \in C} = \varnothing$. For each $(s, \mathcal{C}) \in E_c$, the solution $s \in S_c$ is prohibited by $\mathcal{C}$.

(Step 1) If all variables are assigned, *a solution has been found*, otherwise pick a constraint $c \in C$ with at least one undefined variable $v \in V_c$. Update E_c considering all solutions $s \in S_c$ not already eliminated. (Step 2) If at least one $s \in S_c$ has not been eliminated by E_c, assign s and go to Step 1. (Step 3) Let $s' \in S_{c'}$ be the values most recently assigned, with c' in E_c. If c' cannot be found, *no solution exists*. Otherwise, unassign s', and remove every $(s, \mathcal{C})$ from every $E_{c \in C}$ when $c' \in \mathcal{C}$. Add $(s', \mathcal{C}_{E_c})$ to $E_{c'}$, where $\mathcal{C}_{E_c}$ is the set of all constraints in E_c, and go to Step 1.

Table 1. Steps taken to find a solution with DBT (left) and CD-DBT (right)

Step	1	2	3	4	5	6
α	$\mathbf{1}$	1	1	1	$\mathbf{2}$	2
E_α	$\boldsymbol{\varnothing}$	$\varnothing$	$\varnothing$	$\varnothing$	$\mathbf{1}_\varnothing, \mathbf{34}_\beta$	$1_\varnothing, 34_\varnothing$
β	–	$\mathbf{3}$	3	3	3	3
E_β	$\varnothing$	$\mathbf{124}_\alpha$	124_α	124_α	$\boldsymbol{\varnothing}$	$\varnothing$
γ	–	–	$\mathbf{1}$	1	1	1
E_γ	$\varnothing$	$\varnothing$	$\mathbf{234}_\beta$	234_β	234_β	234_β
δ	–	–	–	–	–	$\mathbf{4}$
E_δ	$\varnothing$	$\varnothing$	$\varnothing$	$\mathbf{1234}_\alpha$	$\boldsymbol{\varnothing}$	$\mathbf{123}_\alpha$
$\#cc$	0	4	4	4	3	4

Step	1	2	3	4	5
$C_{\alpha\beta}$	$\mathbf{13}$	13	13	$\mathbf{23}$	23
$E_{C_{\alpha\beta}}$	$\boldsymbol{\varnothing}$	$\varnothing$	$\varnothing$	$\mathbf{13}_\varnothing$	$13_\varnothing$
$C_{\beta\gamma}$	–	$\mathbf{31}$	31	31	31
$E_{C_{\beta\gamma}}$	$\varnothing$	$\mathbf{24}_{\mathbf{C}_{\alpha\beta}}$	$24_{C_{\alpha\beta}}$	$\boldsymbol{\varnothing}$	$\varnothing$
$C_{\alpha\delta}$	–	–	–	–	$\mathbf{24}$
$E_{C_{\alpha\delta}}$	$\varnothing$	$\varnothing$	$\mathbf{42}, \mathbf{24}_{\mathbf{C}_{\alpha\beta}}$	$\boldsymbol{\varnothing}$	$\mathbf{42}_{\mathbf{C}_{\alpha\beta}}$
$\#ss$	0	2	2	1	2

Example

Suppose we are to solve a CSP where $V = \{\alpha, \beta, \gamma, \delta\}$, $D_{v \in V} = \{1, 2, 3, 4\}$, $C_{\alpha\beta} = \{(1,3),(2,3)\}$, $C_{\beta\gamma} = \{(3,1),(2,4)\}$, and $C_{\alpha\delta} = \{(4,2),(2,4)\}$. For this example, we have chosen to count the number of times a constraint was checked (DBT) or a solution for it was tested (CD-DBT), to give us a rough idea of their performance. DBT performed 19 while CD-DBT performed only 7. The steps are illustrated in Table 1. Note that 124_α (for E_β, step 2) means that β cannot take any of the values $\{1, 2, 4\}$ because of a conflict with the current (at step 2) assignment to α; $13_\varnothing$ (for $E_{C_{\alpha\beta}}$ step 4) means that (α, β) cannot be assigned $(1, 3)$ due to an irresolvable conflict between $C_{\alpha\beta}$ and some other constraint (in this case $C_{\alpha\delta}$).

Experimental Setup

We have produced an initial version of a Constraint Directed Search Library (CDSL) which was inspired by van Beek's constraint library but adds constraint directed search algorithms and the ability to deal with non-binary constraints. We have also developed a random CSP generator whose parameters can be configured to generate CSPs with certain desired properties. We will use these tools to investigate conditions under which CD-DBT offers improved performance. For instance, we are able to set the solution density or the tightness of the constraints in the generated problems. We can then compare the performance of CD-DBT for problems with certain characteristics.

Anticipated Results

While the example we provided was selected to illustrate the potential of CD-DBT, we anticipate that our experiments will confirm the hypothesis that CD-DBT outperforms both CDBT and DBT. We expect the results will be most significant for problems with tight constraints (which can cause thrashing) and for problems with non-binary constraints (where CDBT tends to outperform BT). Both CDSL and our CSP generator will be made available so others can replicate our experiments, and use and extend our work.

References

1. Ginsburg, M. L.: Dynamic backtracking. Journal of Artificial Intelligence Research, Vol 1. (1993) 25–46
2. Pang, W., Goodwin, S. D.: Constraint-directed backtracking. Proceedings of the 10th Australian Joint Conference on Artificial Intelligence. (1997) 47–56

Scheduling Using Constraint-Directed Search

Robert Price and Scott D. Goodwin

School of Computer Science
University of Windsor
Windsor, Ontario
{price9|sgoodwin}@uwindsor.ca

1 Introduction

Here we consider representing Job-Shop Scheduling Problems (JSSPs) as Constraint-Satisfaction Problems (CSPs) and employing constraint-directed search to find optimal or near optimal solutions. A CSP is a problem with a finite set of variables (each with a finite domain of values) and a set of constraints that restrict the possible assignments of values to variables. Scheduling a sequence of jobs involves assigning start times to each job subject to certain constraints. Usually a schedule is sought that minimizes cost and/or some measure of time, like the completion time or makespan.

2 Background

In [3] the job-shop scheduling problem is defined as one where "we are given a set of jobs and a set of machines. Each machine can handle at most one job at a time. Each job consists of a chain of operations, each of which needs to be processed during an uninterrupted time period of a given length on a given machine. The purpose is to find a schedule, that is, an allocation of the operations to time intervals on the machines, that has minimum length". One way of modeling a JSSP as a CSP is given in [2] by having the variables represent the start times of the different activities. The constraints consist of precedence constraints (activity A must end before activity B starts), due-date constraints (the last activity in each job must end before the global due-date), and resource constraints (activities A and B cannot execute at the same time).

In general, a job shop having m machines and n jobs can have a total number of schedules as high as $(n!)^m$. However, more solutions in job-shop scheduling are ruled out by constraint interaction than, for example, in graph colouring [2]. This supports the notion that, as compared to graph colouring, more of the difficulty in job-shop scheduling comes from the interactions of the constraints. We believe that a constraint directed search algorithm could use this tight interaction between constraints to efficiently guide the search and reduce the search space. Thus constraint-directed search should prove valuable in solving job-shop scheduling problems.

One of the simplest constraint-directed search algorithms is Constraint Directed Backtracking (CDBT) [3]. One way to understand the CDBT algorithm is by viewing it as simple Backtracking (BT) in the dual representation of the CSP's underlying constraint graph. In the dual representation, the dual variables are the CSP's constraints, the domains of the dual variables are the tuples that satisfy the constraints, and the dual constraints are compatibility constraints between dual variables that ensure that tuples assigned to dual variables agree with respect to the variables shared by their corresponding constraints. The advantage of backtracking in the dual representation is that the values assigned to variables are drawn from the constraints rather than the domains. In effect, the constraints guide the search. Work on CDBT suggests it outperforms BT when the constraints are tight. Hence, it might be useful for the JSSP.

3 Experimental Setup

In 1963, Fisher and Thompson generated some benchmark problems that have been thoroughly analyzed. Since many have been solved optimally, we will use them to compare CDBT with BT. We will formulate these job-shop scheduling problems as CSPs, and use BT and CDBT to solve them. We will compare the number of nodes visited, number of constraint checks, and runtime to see if CDBT provides an improvement over BT. We will also compare the performance of CDBT with other algorithms that perform forward checking and ones that maintain arc consistency (MAC).

4 Anticipated Results

Although simple CDBT may not perform as well as state of the art algorithms and local search methods when solving job-shop scheduling problems, if it performs better than BT, it would then make sense to investigate enhancements to CDBT such as adding forward checking. If it turns out to be the case that CDBT performs significantly better than BT in this experiment, we can perform other research in the future to see if we can use constraint-directed search methods to reduce the search space when solving job-shop scheduling problems and other highly-constrained CSPs. Furthermore, if constraint-directed search proves valuable in global solving algorithms, our long range goal would be to adapt constraint-directed techniques for local search algorithms and ultimately improve upon the currently best approaches.

References

1. Beck, J. Christopher, and W. Ken Jackson. Constrainedness and the Phase Transition in Job Shop Scheduling, Technical Report TR 97-21, School of Computer Science, Simon Fraser University (1997)
2. Pang, Wanlin, and Scott D. Goodwin. Application of CSP Algorithms to Job Shop Scheduling Problems, The 2nd International Symposium on Operations Research and Its Applications (1996).
3. Vaessens, R.J.M., E.H.L. Aarts, and J.K. Lenstra. Job Shop Scheduling by Local Search, COSOR Memo. 94-05, Eindhoven University of Technology (1994)

Extending Montague Semantics for Use in Natural-Language Database-Query Processing

Maxim Roy and Richard Frost

University of Windsor, 401 Sunset Ave. Windsor Ontario N9B3P4
{royd,Richard} @uwindsor.ca

Montague's semantics has been used in the past for constructing natural-language processors in higher-order functional languages. This paper describes the work done and progress so far in extending a Montague–like compositional semantics in constructing natural-language processors to accommodate n-ary transitive verbs for n>2. In the early seventies, Richard Montague [5] developed an approach to the interpretation of natural language in which he claimed that we could precisely define syntax and semantics for natural languages such as English. Montague was one of the first to develop a compositional semantics for a substantial part of English. By compositional semantics we indicate that the meaning of a compound sentence is determined by the meanings of its constituents and the way they are put together to form sentences. In order to build natural-language query processors, Montague's approach is suitable as it is highly orthogonal. However, it is necessary to extend Montague Semantics in order to build natural-language interfaces that can accommodate more complex language structures.

Higher-order functional-programming languages have been used in the past for constructing natural-language processors based on Montague Semantics. Frost and Launchbury [2] illustrated the ease with which a set-theoretic version of a small first-order subset of Montague Semantics can be implemented in this way. Lapalme & Lavier [4] showed how a large part of Montague Semantics could be implemented in a pure higher-order functional programming language. Frost & Boulos [3] implemented a compositional semantics, based on a set-theoretic version of Montague semantics, for arbitrarily-nested quantified and negative phrases. The approach is based on an extended set theory in which 'negative' phrases that denote infinite sets are represented in complement form.

A comprehensive survey of research on compositional semantics for natural-language database queries has identified little work that has been done in compositional semantics for natural-language queries that includes 3-or-more place transitive verbs.

Currently, in Frost's [3] approach, 2-place transitive verbs are defined as follows (in a simple semantics that doesn't accommodate negation):

```
    verb p = [x | (x, image_x) ← collect verb_rel; p image_x ]
```

Here transitive verbs do not denote relations directly, rather a transitive verb is implemented as function, which takes a predicate on sets as argument. Applying this function to a particular predicate returns as a result a set of entities, which are in the result set if the predicate is true of the entity's image under the associate relation. For example:

```
orbits mars = [x|(x,image_x)←collect orbit_rel; mars image_x]
mars s       = True, if "mars" is_a_member_of  s
             = False, otherwise
orbit_rel = [("luna","earth"), ("phobos","mars"),
             ("deimos","mars")("earth","sol"),("mars","sol")]
```

The definition of `orbits` uses a programming construct called a list comprehension. The general form of a list comprehension is: `[body | qualifiers]` where each qualifier is either a generator, of the form `var ← exp` or a filter, which is a Boolean expression used to restrict the range of the variables introduced by the generators. The collect function used in the above definition of transitive verb is as follows:

```
collect   []       =   []
collect ((x,y):t)  =  (x,y:[e2|(e1, e2) ← t; e1 = x]):
                         collect [(e1,e2)|(e1,e2) ← t;e1 ~= x]
```

By applying collect to the relation orbit_rel, the following is obtained:

```
collect orbit_rel =[("luna",["earth"]),("phobos", ["mars"]),
                    ("deimos",["mars"]),("earth",   etc.
```

So, the final result will be as follows:

```
orbits mars = [x|(x,image_x) ← [("luna",["earth"]),
                                ("phobos", ["mars"]),
                                  etc. ; mars image_x]
            => ["phobos", "deimos"]
```

In order to extend Montague semantics to handle n-place transitive verbs (n >2), we can modify the definition of transitive verbs as follows:

```
  verb p t [x|(x,image_1_x,image_2_x) ← new_collect verb;
                                  p image_1_x ; t  image_2_x]
```

This will allow us to accommodate phrases such as "discovered phobos in 1873".

```
      discover_rel=[("hall",   "phobos",1873),
                   ("galileo","europa",1820),etc.
```

Currently we are developing an appropriate definition for `new_collect` in order to handle n-ary transitive verbs.

Acknowledgements. The authors acknowledge the support of NSERC

References

1. Dowty, D. R., Wall, R. E. and Peters, S. (1981) *Introduction to Montague Semantics*. D. Reidel Publishing Company, Dordrecht, Boston, Lancaster, Tokyo.
2. Frost, R. A., Launchbury, E. J. S. (1989) Constructing natural language interpreters in a lazy functional language. *The Computer Journal - Special edition on Lazy Functional Programming,* 32(2) 108 –121.
3. Frost, R. A., Boulos, P. (2002) An Efficient Compositional Semantics for Natural Language Database Queries with Arbitrarily-Nested Quantification and Negation, 15th Conf. Of the Canadian Society for Computational Studies of Intelligence, AI2002, pp, 252-268
4. Lapalme, G. and Lavier, F. (1990) Using a functional language for parsing and semantic processing. Publication 715a, Department d'informatique et recherché operationelle, Universite de Montreal.
5. Montague, R. (1974) in Formal Philosophy: Selected Papers of Richard Montague, edited by R. H. Thomason. Yale University Press, New Haven CT.

An Investigation of Grammar Design in Natural-Language Speech Recognition

Yue Shi and Richard Frost

School of Computer Science, University of Windsor, 401 Sunset Ave. Windsor ,ON N9B3P4
{shic, richard} @uwindsor.ca

Accuracy and robustness are two competitive objectives in natural-language speech-recognition. How to achieve good accuracy and good robustness is a significant challenge for speech-recognition researchers.

Stochastic (statistical) techniques and grammar-based techniques are the two main types of speech-recognition technology. A Statistical Language Model (SLM) is simply a probability over all possible sentences or word combinations, whereas grammar-based techniques use grammars to guide the speech-recognizer in a particular application. Statistical language models were popular around 1995, whereas grammar-based language models took the pre-eminent position in commercial products by 2001.

In order to investigate the effect of grammar design on accuracy and robustness, we conducted an experiment using three types of recognition grammar using a commercial off-the-shelf VXML speech browser.

1) A simple *word-sequence grammar* defining all sequence of words from the dictionary up to length ten. For example:
2) `<wordsequence>=<word>|<word><word>|etc.(to length 10)`
3) A *syntactic grammar* consisting of rules defining all syntactically-correct expressions in the inpouyt language: For example:
4) `<synquestion> = (which) <nounphrase><verbphrase>;`
5) *A Semantic Grammar***.** Based on the observation that some syntactically correct utterances may semantically wrong, some semantic constraints can be directly encoded to syntax to exclude utterances such as "which man orbits mars?" A sample semantic grammar, which requires that an animate (/inanimate) noun phrase be followed by an animate (/inanimate) verb phrase, is as follows:

```
<semquestion>
    =(which)<animate_nounphrase><animate_verbphrase>
    |(which)<inanimate_nouphrase><inanimate_verbphrase>;
```

The following table shows the size and branching factors of the languages defined by the three grammars used in our experiment:

Grammar	Language size	Branching factor
Word sequence	$2.40 * 10^{27}$	547
Syntactic	$8.17 * 10^{15}$	267
Semantic	$5.55 * 10^{12}$	96

Two users were involved in the experiment: a male native-English speaker and a female non-native English speaker. Three sets of testing utterances were used: (1) a semantic set, which included queries that are both semantically correct and syntactically correct, (2) a syntax set, which included queries that are only syntactically correct but semantically incorrect, and (3) a word-sequence set, which covered only word sequences that were neither semantically correct nor syntactically correct. Recognition results were recorded as : "correct recognition", "incorrect recognition" (misrecognition), and "not recognized" at all. Each of the two users spoke several hundred utterances from the three sets using the three different recognition-grammars. The results of the experiment are summarized as follows:

Type of query	Recognition Grammar	Correctly Recognized %	Incorrect%	Not recognized %
Semantically correct	word seq.	12	60	28
	syntactic	66	14	20
	semantic	75	4	21
Syntactically correct	word seq.	8	44	48
	syntactic	65	5	30
	semantic	0	22	78
neither	word seq.	15	56	29
	syntactic	0	30	70
	semantic	0	10	90

The results are much as expected. However, there were three unexpected results:

1) The recognition rates for the syntactic and semantic grammars were unexpectedly high given the sizes of the languages, which is a good indication of the value of grammar design in commercial speech-recognition products.
2) The correct-recognition rate did not improve significantly for the semantic grammars with semantic queries compared to the syntactic grammars, but the misrecognition rate went down significantly, which is a good indicator for the value of semantic constraints.
3) The word-sequence grammars were better at spotting words than expected, given the very high branching factor, which is a good indicator for its inclusion in a future "combined grammar" approach.

These experimental results confirm that recognition accuracy can be improved by encoding semantic constraints in the syntax rules of grammar-driven speech recognition. However, the results also indicate that grammar design is a complex process that must take into account application-specific requirements for accuracy and robustness. Also, the results suggest that it is worth investigating the use of combined "weighted" recognition grammars. The most-constrained semantic grammar could be applied first (having the highest weight) and if it fails to recognize the input, then the syntactic grammar is tried, followed by the word-sequence grammar if the syntactic grammar fails. This is the focus of future research.

This work was supported by a discovery grant from the Natural Science and Engineering Council of Canada.

Genetic Algorithm Based OSPF Network Routing Using LEDA

Lenny Tang, Kay Wiese, and Vive Kumar

InfoNet Media Center
Simon Fraser University, Surrey, Canada, V3T 2W1
lctang@sfu.ca, wiese@sfu.ca, vive@sfu.ca

As a predominant technique to estimate the cause and effect of Quality of Service (QoS) criteria in computer networks, simulations provide insight into how to most efficiently configure protocols to maximize the usage and to estimate the criteria for acceptable performance of network applications. This paper investigates the simulation of Genetic Algorithm-based network routing using Open Shortest Path First (OSPF) protocols. The simulator is implemented using LEDA (Library of Efficient Data types and Algorithms) [1] and the applicability of the library is examined in the context of Genetic Algorithms.

OSPF is the most widely used routing protocol. OSPF's popularity comes from its scalability, efficiency, and its self-sufficient nature. In supporting hierarchies and autonomous areas, OSPF allows for networks to grow and still be easily maintainable. By allowing the administrators to set the cost value for a network link between two routers, OSPF can intelligently route packets along links the administrators deem to be better. By broadcasting Link-State Advertisements (LSAs), routers can be added and removed, and OSPF would regenerate the routing tables automatically to reflect the changes in the network. OSPF was designed specifically to use Dijkstra's Shortest Path First algorithm. Dijkstra's algorithm determines the shortest routes to all of the routers in the area using the LSA database. By adjusting the costs (weights) of the links one can alter the routes that Dijkstra's algorithm will calculate.

In an attempt to optimize the costs associated to the link in OSPF, Fortz and Thorup [2] proposed a local search heuristic, HeurOSPF, to set the weights in a simulated network demand set. The results achieved in their HeurOSPF compared quite well to the other heuristics (e.g., InvCapOSPF, UnitOSPF, RandomOSPF, and L2OSPF), and relatively close to the optimal solution. Expanding on HeurOSPF, GAOSPF [4], using a genetic algorithm, estimates the candidate cost sets.

Using genetic algorithms in this domain is a relatively new trend, which is necessitated by the inherent nature of certain protocols that deal with NP-hard problems. Also, it has been shown that weight setting for a given demand set is NP-hard [2]. Weight setting is the process of applying a cost value to the links in a network. In the case of OSPF, once these costs are applied, Dijkstra's algorithm can be used at each router to determine the shortest path with the lowest total cost to each of the routers in the OSPF area. In GAOSPF, the chromosome represents the links in a network. In other words, each gene in the chromosome corresponds to a specific link in the network. The fitness for a chromosome is calculated by first applying Dijkstra's algorithm at each router using the chromosome for the cost, so that a next-hop routing table for each node can be formed. Then, simulated demands on the network are "routed" according to the routing tables just created, and the link bandwidth usage is recorded for each link.

By adjusting the weights in OSPF we are able to change the routing tables generated by Dijkstra's algorithm. In the simulation, the load demands that are to be routed are predefined. GAOSPF routes these demands on the network for each member of the population. The fitness function calculates a fitness value for the chromosomes in the population and defines the traits in the chromosomes that GAOSPF is trying to groom. The chromosomes are then ranked and mated. This process is repeated for a number of generations in hopes that the chromosomes become fitter.

Network routing can be represented by a weighted graph problem. A network consists of routers that are connected by links. LEDA provides a convenient way to assign values to nodes and edges in a graph. LEDA also has a group of graph algorithms that are ready for use with its data types. Our implementation of GAOSPF used LEDA data structures and Dijkstra's algorithm to implement a GA-based network routing simulator for OSPF protocols. In our implementation of GAOSPF, the five main data types we used were *graphs*, *nodes*, *edges*, *node arrays* and *edge arrays*. The graphs, nodes, and edges represent the network topology, routers, and the links respectively. Using these data structures from LEDA, we implemented GAOSPF by building required functions such as: the Crossover function, Routing Table function, and the Fitness function. Of particular importance to the OSPF protocol is the Dijkstra's algorithm. Given an edge array storing the weights for a graph and a source node, Dijkstra's algorithm will return a predecessor node array. The predecessor node array is similar to a routing table in that it contains an entry for each node. The edge arrays are convenient data types to represent chromosomes in GAOSPF, and as exemplified in the prototype, it is possible to use edge arrays in other GA problem domains.

One of the promising extensions of our research is to examine the merits of applying different crossover and recombination techniques to improve the results produced in GAOSPF. Once a set of crossovers, recombination techniques, and parameters have been found to yield near optimal solutions, a study can be conducted on the routing table changes for different demands to predict changes for a given set of weights in OSPF. This can be used to regulate network routing that is adaptive to different types of demands on the network.

References

1. The Library of Efficient Datatypes and Algorithms (LEDA). Algorithmic Solutions Software GmbH. Available from: http://www.algorithmic-solutions.com
2. B. Fortz & E. Thorup. "Internet traffic engineering by optimizing OSPF weights" in Proc. IEEE INFOCOM, (2000) 519-528
3. K.G. Coffman, & A.M. Odlyzko. "Internet growth: Is there a "Moore's Law" for data traffic?", J. Abello, P.M. Pardalos & M.G.C. Resende (Eds.), Kluwer Academic Publishers, (2001) 47-93
4. M. Ericsson, M.G.C. Resende, & P.M. Pardalos. "A genetic algorithm for the weight setting problem in OSPF routing". Journal of Combinatorial Optimization, Vol 6 (3). Kluwer Academic Publishers, (2002) 299-333

A Multi-agent System for Semantic Information Retrieval

Yingge Wang and Elhadi Shakshuki

Jodrey School of Computer Science, Acadia University, Nova Scotia, B4P 2R6
{050244w, elhadi.shakshuki}@acadiau.ca

Abstract. This paper presents an ontology-based multi-agent information retrieval system architecture. The aim of this system is to help Web users gather the right information based on the interpretation of the Web at the semantic level. The system consists of different types of agents that work together to find and process information and disseminate it to Web users in semantically encoded form. Interface agents interact with users, receive users' queries and display results. Semantic agents extract the semantics of information retrieved from web documents, and provide the knowledge in ontologies to other agents. Service agents manage the ontology-knowledge base. Resources agents locate and retrieve the documents from distributed information resource. To demonstrate the facility of the proposed system, a prototype is being developed using Java, Zeus toolkit [5] and Protégé-2000 [6].

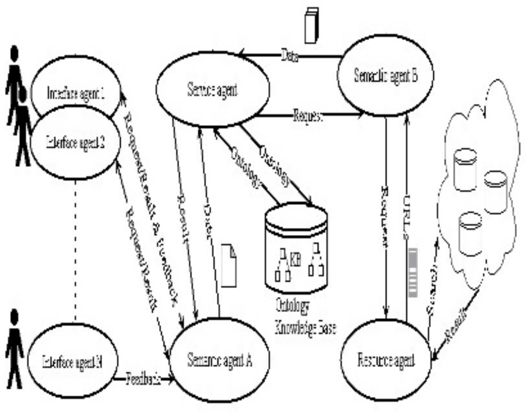
1 Introduction

The process of finding and searching the desired information from the huge body of the Web of information has become a time-consuming task more than ever. Today, Web users face the challenge of locating, filtering and organizing the ever-changing and ever-increasing information from distributed heterogeneous resources. A "one-for-all" search engine would not satisfy millions of people, each with different backgrounds, knowledge and needs.

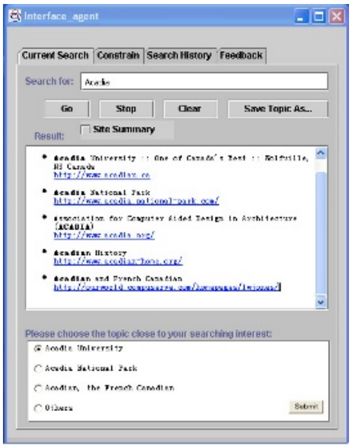
Software agents exhibit dynamic and flexible character, including autonomy, pro-activity, adaptation and social ability [1]. In this work, we proposed ontology-based multi-agent information retrieval system that allows users to submit queries and get the desired results through a graphical user interfaces. The agents of the system communicate with each other using KQML like language [2]. The system allows the users to provide feedbacks to improve the quality of the search. When the system receives a query, it parses and processes the query to provide the relevant information, utilizing Web resources. The results are based on the semantic of the information, and the understanding of the meaning of the user's request and feedback. An ontology-knowledge base is used as a virtual representation of the user's knowledge. The proposed system incorporates several techniques, including semantic extrication, topic extraction, and ontologies comparison.

2 System Architecture

The main objective of this work is to develop an ontology-based multi-agent information retrieval system, which could interpret the Web at the semantic level, and retrieve the information based on the users' interests. Figure 1-a shows the architecture of the proposed system. The interface agent allows the user to interact with the environments through the use of natural language. It provides a graphical user interface (GUI) for the user to submit queries with the desired constrains, to provide feedbacks, and to display results. The semantic agent's main responsibility is to interpret concepts, relations and information using a semantic wrapper [3], and send results to the service agent. The semantic agent ensures a mutual understanding between the user and the Web on the semantic level. Information extraction and well-defined concepts are the fundamental components for the semantic agent. The service agent acts as a service provider that pairs semantic agents according to users' interests. When the service agent gets results back from the semantic agent, it compares these results in terms of ontology and the relevance of the retrieved document. By doing so, it attempts to find documents that have potential to answer the user's query. The service agent is the central point for all agents' activities and information processes, because it maintains the ontology-knowledge base, which stored document relevancy data and websites evaluation history. The ontology-knowledge base contains terminological knowledge exploited by the service agent and the semantic agent. This knowledge structure is provided by WordNet® [4], which is an online lexical reference system. The resources agent acts as a search tool that provides access to a collection of information sources. It traverses the Web to its designated location. When the resources agents identify a document that contains the required topic or synonyms, they pass the URL to the semantic agent to exam if it fulfills the user's predefined search constrains, such as published date, file format, written language, file size, etc. The URLs that do not satisfy search constrains will be removed from the result list, and the rest will be passed on to the service agent. An example of interface is shown in Figure 1-b.



(a)



Search for: Acadia

- **Acadia** University :: One of Canada's Best :: Wolfville, NS Canada
  http://www.acadiau.ca
- **Acadia** National Park
  http://www.acadia.national-park.com/
- Association for Computer Aided Design in Architecture (**ACADIA**)
  http://www.acadia.org/
- **Acadian** History
  http://www.acadian-home.org/
- **Acadian** and French Canadian
  http://ourworld.compuserve.com/homepages/lwjones/

Please choose the topic close to your searching interest:

[✓ Acadia University]
Acadia National Park
Acadian, the French Canadian
Others

(b)

Fig. 1. (a) System architecture and (b) An example of an interaction window.

References

[1] Genesereth, M.R.and Ketchpel. S.P., “Software Agents”, *Communications of the ACM, Vol. 37(7),* pp. 48-53, 1994.

[2] Finin, T., Labrou, Y. and Mayfield, J., ‘‘KQML as an Agent Communication Language’’, In Bradshaw J.M. (Ed.) Software Agents, Cambridge, MA: AAA/MIT Press, pp. 291-316, 1997.

[3] J.L. Arjona, R. Corchuelo, A.Ruiz and M.Toro. “A knowledge Extrication Process Specification”, in Proceedings of *IEEE/WIC International Conference on Web Intelligence,* pp.61-67, 2003.

[4] WordNet®, http://www.cogsci.princeton.edu/~wn/

[5] Zeus toolkit, http://www.btexact.com/projects/agents/zeus/

[6] Protégé-2000, http://protege.stanford.edu/index.html

Decision Mining with User Preference

Hong Yao

Department of Computer Science, University of Regina, SK, Canada, S4S 0A2
{yao2hong }@cs.uregina.ca

1 Statement of Problem

We are drowning in information, but starving for knowledge. It is necessary to extract interesting knowledge from large and raw data. Traditional data mining approaches discover knowledge based on the statistical significance such as frequency of occurrence, which leads a large number of highly frequent rules are generated. It is a tough work for users to manually pick the rules that they are really interested. It is necessary to prune and summarize discovered rules. Most importantly, different users have different interestingness in the same knowledge. It is difficult to measure and explain the significance of discovered knowledge without user preference. For example, two rules Perfume → Lipstick and Perfume → Diamond may suggest different potential profits to a sales manager, although both are frequent rules.

It motivates us to allow users to express their preference into the knowledge mining process, because only users know exactly what they demand. User preference should be a important utility to measure knowledge. Furthermore, our proposed approach emphasizes that the discovered knowledge should have the ability to suggest profitable actions to users according to the user preference, because the discovered knowledge is useful only it can be used in decision-making process of the users to increase utility. Thus, our proposed approach discovers high utility knowledge regard as user preference, decisions are made based these high utility knowledge. The decisions, which can increase user utility, are recommended to users. The quality of the decisions is guaranteed by the utility of decisions.

2 Principle of Decision Mining with User Preference

Our proposed approach emphasizes the significance of user preference to decision making. The proposed decision mining model consists of probability mining model and utility mining model. Probability mining model associates each candidate itemset with a probability, the percentage of all transactions that contain the itemset. The possibility of itemset can be mined directly from the transaction table using the traditional Apriori algorithm. Utility mining model quantifies each candidate itemset with a utility, the sum of the utilities of its items. In this paper, the user preference is regarded as a utility and expressed by the associated utility table. The value of utility reflects the impact of user preference on the items. It can be mined by incorporating the utility table with the transaction table. The possible action consequences are quantified by combining the possibility and utility of itemset together. Thus, each discovered knowledge has assigned a utility. Decisions are made on the basis of maximum expected utility of discovered knowledge. In other words, decision knowledge is the knowledge with the most highest expected utility based on the probability and utility of itemset. The advantages of decision mining with user preference is that the goal of data mining can be guided by the user preference. The user objectives can be maximum, and the most interested knowledge, which has the highest utility, will be recommended to users.

Our approach consists of four parts. First, the user preference is formally defined as a utility table. If users prefer item a to item b, the utility of a will be higher than which of b. Next, push utility table inside mining process. The possibility and utility of itemset can be mined simultaneous by integrating the utility table and the transaction table together. In order to reduce search space, and prune uninteresting itemset, the potential mathematic properties of utility are analyzed, the utility boundary property and possibility boundary property of itemset are demonstrated. A heuristic approach is proposed to calculate the expected utility of itemsets. Only itemsets with high expected utility are kept. Third, the detailed algorithm is designed according to the theoretic model. By applying these properties, an upper bound on the utility of a $k-$itemset can be calculated by analyzing the utilities of the already discovered $(k-1)-$itemsets. Finally, the system is implemented, and the effectiveness of the approach is examined by applying it to large synthetic and real world databases.

3 Experiment Results

We used a variety of synthetic datasets generated by the IBM synthetic data generator. We report our results for a single synthetic dataset of 9 million records with 23,840 items. 39 highest utility knowledge are discovered when threshold $\alpha = 0.10$. There are about eight million records in our real world databases. The size of database is about 651M. The transaction table represents the purchase of 2238 unique items by about $428,665$ customers. 18 highest utility knowledge is discovered when threshold $\alpha = 0.10$.

4 Conclusion and Potential Applications

The discovered knowledge is considered useful or interesting only it conforms to the user preference. The ignored user preference should play an important role in knowledge discovery. The solution of this problem will contribute to many applications such as recommend system, web mining, information retrieval, and collaborative filtering, because utility also exists such applications. For example, if the goal of a supermarket manager is to maximize profit, it is not enough to make decision based on only the transaction database. Profit is a utility. It is necessary to incorporate user preference into the mining process.

Coarsening Classification Rules on Basis of Granular Computing

Yan Zhao

Department of Computer Science, University of Regina,
Regina, Saskatchewan, Canada S4S 0A2
yanzhao@cs.uregina.ca

1 Problem Statement

The task of classification, as a well-studied field of machine learning and data mining, has two main purposes: describing the classification rules in the training dataset and predicting the unseen new instances in the testing dataset. Normally, one expects a high accuracy for precisely descriptive use. However, a higher accuracy is intended to generate longer rules, which are not easy to understand, and may *overfit* the training instances. This motivates us to propose the approach of coarsening the classification rules, namely, reasonably sacrifice the accuracy of description in a controlled level in order to improve the comprehensibility and predictability. The framework of granular computing provides us a formal and systematic methodology for doing this. In this paper, a modified PRISM classification algorithm is presented based on the framework of granular computing.

2 The Formal Concepts of Granular Computing

The formal general granular computing model is summarized in [2]. Briefly, all the available information and knowledge is stored in an *information table*. The *definable granules* are the basic logic units for description and discussion use. The *refinement*, or *coarsening* relationship between two definable granules is a partial order, which is reflexive, asymmetric and transitive. For the tasks of classification, we are only interested in *conjunctively definable granules*, which mean that more than one definable granule are conjunctively connected. *Partition* and *covering* are two commonly used granulations of universe. One can obtain a more refined partition by further dividing equivalence classes of a partition. Similarly, one can obtain a more refined covering by further decomposing a granule of a covering. This naturally defines a refinement (coarsening) order over the partition lattice and the covering lattice.

Approaches for coarsening decision trees include pre-pruning methods and post-pruning methods. For pre-pruning methods, the stopping criterion is critical to the classification performance. Too low a threshold can terminate division too soon before the benefits of subsequent splits become evident; while too high a threshold results in little simplification. Post-pruning methods engage a non-trivial post process after the complete tree has been generated. Pre-pruning methods are more time-efficient than post-pruning methods. Approaches for coarsening decision rules only include pre-pruning methods, which have the same advantages and disadvantages as they are used for decision trees.

3 The Modified-PRISM Algorithm

The PRISM algorithm is proposed as an algorithm for inducing modular rules [1] with very restrictive assumptions. We modify the PRISM algorithm to get a set of coarsening classification rules by using a pre-pruning method.

```
For each value $d$ of the decision attribute:
1. Let $\chi$ = “”.
2. Select the attribute-value pair $\alpha_x$ for which $p(d|\alpha_x)$ is the maximum.
   Let $\chi = \chi \wedge \alpha_x$. Create a subset of the training set comprising all the
   instances which contain the selected $\alpha_x$.
3. Repeat Steps 1 and 2 until $p(d|\chi)$ reaches the threshold, or no more
   subsets can be extracted.
4. If threshold = 1
   4.1: Remove the $d$-labeled instances covered by $\chi$ from the training set.
   4.2: Repeat Step 1-4.1 until all $d$-labeled instances have been removed.
  Else      /* for coarsened rules
   4.1’: Remove the attribute-value pairs used for $\chi$ from consideration.
   4.2’: Repeat Step 1-4.1’ until no more attribute-value pairs are left.
```

In Step 2, if there is only one α_x for which $p(d|\alpha_x)$ is the maximum, then α_x is of course selected. If there are more than one α_x that is the maximum at the same time, then the one, conjuncted with the previous χ, that covers a larger number of instances is selected. When the accuracy threshold is one, then the original PRISM algorithm is applied, (Step 4.1 and 4.2). When the accuracy threshold is less than one, we cannot simply remove the d-labeled instances from the training set. Instead, we remove the “used” attribute-value pairs from the consideration. This greedy method is stated in Step 4.1’ and 4.2’.

4 Conclusion

The purposes of coarsening classification rules are for a better understanding of the classification rules and a higher accuracy for prediction. By putting the classification in the granular computing framework, we can study the problem formally and systematically. We modify the existing PRISM algorithm to coarsen the classification rules.

References

1. Cendrowska, J. PRISM: An algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, **27**, 349-370, 1987.
2. Yao, Y.Y. and Yao, J.T. Granular computing as a basis for consistent classification problems, *Proceedings of PAKDD’02*, 101-106, 2002.

Author Index